JIMMA UNIVERSITY

COLLEGE OF NATURAL SCIENCE

DEPARTMENT OF INFORMATION SCIENCE

DEVELOPING A KNOWLEDGE BASED SYSTEM TO SUPPORT DEBUGGING COMPUTER PROGRAM SOURCE CODE

By: Tariku Fetene

June, 2017

Jimma, Ethiopia

JIMMA UNIVERSITY

COLLEGE OF NATURAL SCIENCE

DEPARTMENT OF INFORMATION SCIENCE

DEVELOPING A KNOWLEDGE BASED SYSTEM TO SUPPORT DEBUGGING COMPUTER PROGRAM SOURCE CODE

A THESIS SUBMITTED TO COLLEGE OF NATURAL SCIENCES OF JIMMA UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF INFORMATION SCIENCE (INFORMATION AND KNOWLEDGE MANAGEMENT)

By: Tariku Fetene

June, 2017

Jimma, Ethiopia

JIMMA UNIVERSITY

COLLEGE OF NATURAL SCIENCE

DEPARTMENT OF INFORMATION SCIENCE

DEVELOPING A KNOWLEDGE BASED SYSTEM TO SUPPORT DEBUGGING COMPUTERPROGRAM SOURCE CODE

By: Tariku Fetene

Name of Member of the Examiner Board

Name	**Title**	**Signature**	**Date**
___________	Chairperson	___________	___________
___________	Advisor	___________	___________
___________	Co - Advisor	___________	___________
___________	External Examiner	___________	___________
___________	Internal Examiner	___________	___________

DEDICATION

I dedicate this research to my dearly loved, genuine, very kind, God-fearing, and beautiful and even much more to me, mother who passed away not taking part of my success and happiness with me. You are always in my heart, Rest in peace!

Acknowledgement

First and for most, my great thank goes to God who consent the accomplishment of this research. His mother also deserves a thank. Next, I would like to thank Dr. Million Meshesha and Mr. Miniyechil Belay for their unreserved and genuine comments and advise throughout the improvement of this research. I also express my thank to Mizan Teppi University school of informatics staff members and Jimma University Department of Information Science who provided me with the facilities required for this research.

Afterward, I would like to express my gratitude to all family members of mine who help me in one way or another for the successful accomplishment of this very research. Once again thank you to everyone who never gave up on me, made sure I was on the right track, supported me all throughout this struggle, and believed I can make it!

Table of Contents

List of figures

List of tables

Abbreviations and Acronyms

AI	Artificial Intelligence
DE	Domain Expert
ES	Expert Systems
HC	Harmonized Curriculum
IDE	Integrated Developmental Environment
IE	Inference Engine
ICT	Information and Communication Technology
KB-DAS	Knowledge Base Debugging Assistance System
KBS	Knowledge Based System
KE	Knowledge Engineer
KR	Knowledge Representation
ResQue	Recommender Systems' Quality of user experience
RBS	Rule Based System
MTU	MizanTeppi University

Abstract

To write programming is an essential skill for students of computing and informatics. However, learning programming skills in finding errors and correcting them to make the program run has been recognized as a great challenge for novice programmers. Most of the errors currently thrown by the compiler don't automatically point the novice programmers to the right direction since some of the messages need deep understanding and expertise. This finally leads students to suffer and discouraged hope of solving the problem on their own which results in programming phobia. Even worse, repetitive failures may defeat students' enthusiasm for learning. This study therefore investigates a number of different compilers, logical and run time errors that novice programmer's encounter and the associated debugging behaviors to assists them in writing error free code so that the primary objective of the study is to develop a knowledge based system to assist novice programmers in debugging computer program source code. This research has followed a Knowledge engineering process for knowledge acquisition, modeling, representation and prototype development and testing, The conceptual model of the knowledge based system is designed by using a decision tree structure which is easy to understand and interpret the causes involved in the program error. Based on the conceptual model, the knowledge is represented using 'if – then'rules. The developed prototype infers the rules by backward chaining and provides appropriate suggestions as per the users query. The prototype was evaluated for its usability in which it registers 87.27% user's acceptance. In addition, the performance of the system was evaluated with twenty five test cases. The results of the validation test indicate that the prototype registers on the average of 80% accuracy. Most of the errors handled in this study are compile time errors. Our observation shows that there is a great challenge in understanding logical errors in writing source codes which left as a future research direction.

Keywords: Computer Programming, Debugging, Knowledge Base System

CHAPTER ONE
Introduction

1.1 Background of the study

Many researchers mention that programming is a very useful skill and can be a rewarding career (Kurland *et al*, 2013). The primary mission of each tertiary institution in the field of computing and information technology is to offer high quality and relevant education in order to produce skillful and competent graduates. Akinola *et al*.(2015) mentioned that the students' academic performance is the outcome of the final examinations, quizzes, assignments, attendance and other graded points related to the course. To accomplish this, a number of practical and instructional strategies were designed to improve the students' academic performance in programming (Akinola *et al*., 2015).

According to (Winslow, 2013), programming courses generally contain lots of practical exercises: the issues to be learned do not become concrete for the student until start witting a program. The first programming courses aim at giving students the basic programming skills on which they can later build more advanced skills and knowledge. In practical exercises novice students do errors and challenged for fixing those errors which finally leads students to be stopper, those students who simply stop and abandon all hope of solving the problem on their own while they notice the program has error (Winslow, 2013).

Debugging is a process of removing bugs from coded programs. If the program is not working according to design, developers must debug the source code and fix the issues. Hwang *et al*. (2012) mentioned it has been known to account for more than half of the effort and time spent in software development.

Level of debugging skill is one of the major differences between novice and expert programmers. Experts make fewer errors and locate and correct bugs faster than novices. Debugging training is even more needed by novice programmers (Oman *et al*, 2011). Unlike experienced programmers who can easily locate errors or root causes of a problem, novice programmers often turn to trial and errors for debugging.

Technologies, especially knowledge based system are playing a big role in educational development and for the revolution in learning systems. They bring new opportunities to the educational system (Marcus and McDermott, 2011).The emergence of Knowledge Based Systems (KBS) provides a means for students, doctors, lawyers, engineers and other people to use the computer as an aid in finding at solution for their problem. KBS are interactive computer programs that incorporate the knowledge and judgment of experts in appropriate domains. These systems promise to introduce changes at least as far-reaching as the entire computer revolution to date. The development of a KBS presently involves the cooperative effort between a knowledge engineer (KE) and one or more experts who possess the domain-dependent knowledge. The KE elicits the knowledge and uses either an expert system building tool or a general-purpose language to represent and manipulate it.

1.2 Statement of the problem

Learning computer programming is a complex task since programming requires new ideas in thinking and creative skills in problem solving (Hristova, 2003).Programming is a skill that is considered hard to learn and even after two years of instruction, the level of programming understanding is low (Kurland *et al.*, 2013). However, if supported by suitable teaching strategies and tools it can be mastered by pupils to some extent (Akinola*et al*., 2015).

The need assessment conducted at Mizan Teppi University revealed that 61% of students in the University have poor academic performance in programming courses [*appendix I*]. There observed high dropout rate in programming courses. Since most courses offered in computing and informatics related to programming courses observed a high percentage of students with poor academic performance every semester as noted by Marcus and McDermott (2011), this affect the quality of education greatly.

In the scientific literature, many reasons are pointed out for student’s high dropout rate, such as the following. Students’ motivation, way of study, methodology and tools used traditional teaching methods, normally based on lectures and specific programming language syntaxes, often fail in what concerns the students’ motivation in getting involved in meaningful programming activities (Schulte &Bennedsen, 2006). But are these challenges a reason to achieve low grade (61%) for our students in the local University or are there other challenges in the teaching learning process of computer programming courses? were addressed in this research.

Students from computing and informatics take as many programming course in their stay in the University as a partial fulfillment for their degree. Novice programmers, however, easily forget issues of programming style during programming coursework. In particular in most classes, most students fail in programming courses or else, in some cases, may pass the courses without having a good knowledge. Winslow (2013) noticed that students may know the syntax and semantics of individual statements, but they do not know how to combine these features into valid programs.

Actually high level programming language like turbo C++, Quincy, java tell what and where the problem is but lots of novice students still confused that is why they need their teachers to correct their errors Winslow (2013). Compiler is not helpful all of the time (Salcedo, 2016). It may give misleading messages regarding the error. This can cause confusion to novice programmers. Too often, compile error messages are hidden, long, or hard to understand even for experienced programmers (Kordaki, 2010). That is why some students even can't fix the same errors that already fixed by the instructor and again the question raised by the students may despair instructors. According to Kordaki, (2010) this problem is worse especially in large programming class sizes since it is very difficult and time-consuming issue for teachers to assess and give feedback to students every time. Syntax errors make a program incomprehensible to compilers and are then easily pointed out. While compilers detect the obvious syntax errors, their error messages do not necessarily point the students to the right direction needed to fix errors in the source code (Salcedo, 2016).

Computing curricula rarely provide formal debugging training (Harmonized curriculum, 2013). Novice programmers are then left to develop their own skills. When they do, they develop debugging skills with limited abilities in formulating ideas about the possible bugs in their code. A more recent study, Lewis and Gregg discussed the benefits of introducing certain debugging tools earlier or later in the curriculum (lewis& Gregg, 2016). The observation conducted throughout the research shown that most of novice students are struggling to find and correct their errors by trial and error which takes lots of time and even sometimes lead them into stopping trying and loss hope in programming. Hence, it seems logical to start them early case; those who are trained early in debugging would become better debuggers more quickly. Therefore, it has become an important and challenging issue to develop improvement strategies or tools for assisting novices in debugging experience (Hwang *et al*., 2012).

Therefore, the more specific errors messages from the compiler help students to clarify concepts, misconceptions, or improve their mental models (Salcedo, 2016). In particular, it helps them become better programmers in terms of debugging and writing error-free programs by improving their program comprehension skills and giving them debugging experience for the next programming courses in their stay in the University. Thus, the principal concern of this study is to design a knowledge based system for supporting novice programmers in debugging source code and writing error free program. To the end, the study attempt to explore and answer the following research questions.

1.3 Research questions

The research questions which are explored and answered by this research has presented as follow:

- ❖ What are the challenges, for both teachers and students, in teaching learning computer programming?
- ❖ What are the common errors novices are experiencing and how can these problems be solved?
- ❖ What knowledge of programming style and coding conventions are there to write error free program?
- ❖ To what extent the application of knowledge based system support novice programmers in debugging computer program?

1.4Objectives of the study

1.4.1 General objective

The general objective of this study is to assess the teaching learning process of computer programming and developing a knowledge based system for supporting novice programmers in debugging source code written in high level programming language.

1.4.2 Specific objectives

- To assess the challenges for both teachers and students, in teaching learning computer programming
- To explore the common errors novice programmers are encountered
- To explore the knowledge of programming style and coding conventions in finding and correcting those common errors novices experiencing
- To identify, acquire, model and represent the knowledge required for the knowledge based system development.
- To develop, test and evaluate the prototype knowledge based system

1.5Scope and Limitations of the study

There are a number of different approaches for designing knowledge based system but this system has employed with rule based approach. According to Sajja and Akerkar (2010). Rule based reasoning in the development of knowledge based system is advantageous in compact representation of general knowledge. It is also mentioned as rules can easily represent general knowledge about a problem domain. Rule based representation has uniform syntax. Hence, the meaning and interpretation of each rule can be easily analyzed. Each rule is an independent piece of knowledge about the problem domain. Rules are a very natural knowledge representation method with a high level of comprehensibility. Rules can emulate the expert‘s way of thinking in natural expression (Sajja & Akerkar, 2010). And finally it is founded that errors novice programmers experiencing can easily represented with rule based.

The research is compassed for novice undergraduate students from school of computing and informatics since the negative impact of basic introductory courses may have harmful consequences in the learners' attitude towards the next programming courses and totally to field of study. Those who are trained early in debugging would become better debuggers more quickly (Hwang *et al*, 2012).The research is limited for a specific programming language called C++ since novice programming students are practicing their first program is in C++.The research is ranged on developing knowledge based debugging assistance system prototype according to the common errors which novices are encountering as it is gathered from their instructors.

One of the limitations for this research is extracting the logical errors and codify it into a format that can be used in knowledge based system and then representing it into rule based. So, it could not be handled by the system as many as expected logical errors. There are three common types of computer programming errors but there is no exactly specified number for the errors novice programmer do in each type of errors. The C++ IDE has no additional feature which enables to add an extension into it just like other programming languages; Java and VB. So that with this limitation the researcher forced to develop a standalone system which enhance the C++ compiler by providing a specific error message. Another limitation was giving the knowledge base system self learning ability.

1.6 Significance of the study

Since those who are trained early in debugging would become better debuggers more quickly and it is found as logical to start from novices (Hwang *et al*., 2012), this research would be useful for students in introductory computer programming classes. This improves their program comprehension ability and gives them debugging experience. It also minimizes the time students spend on finding and correcting errors by trial and error. The system can act like their instructor in case of error happening while practicing to write computer program source code by their own since their instructors could not be available 24/7. Instructors are benefited from the system in a way to be assisted in the area that their students needed a help on fixing errors, especially for large class size (which is large number of students in the class). Since compiler designers seem to ignore helping a special group of programmers: novice programmers because they often encounter cryptic compiler error messages that are difficult to understand and thus difficult to resolve, Compiler designers are befitted from this research to think over on developing a more detailed errors notification for novice programmers. Future researchers are benefitted, to study on how to assist students in improving their debugging experiences in computer programming courses.

1.7 Methodology of the study

1.7.1 Research Approach

Research approaches are strategies of inquiry that provide specific direction for procedure in a research design. Creswell (2003) classified scientific research approaches into three: quantitative, qualitative and mixed research. Qualitative research seeks to describe various aspects of social and human behavior through particular methods such as interview, observation, focus of quantitative properties and phenomena and their relationships. Whereas, mixed research approach involves collecting and analyzing both quantitative (numeric) and qualitative (descriptive) forms of primary data in a single study (Creswell, 2003).

Accordingly, this research follows mixed approach in addressing the research question raised. The use of quantitative approach is to evaluate/measure perception of students in the overall teaching learning computer programming. On the other hand, the deployment of qualitative approach; i.e. semi structured interview for teachers is to ascertain or triangulate and complete the information obtained from students and to investigate the overall problems which novice are encountering in the debugging process of computer programs.

1.7.2 Research Design

Over the years, the discipline of knowledge engineering has evolved into the development of theory, methods and tools for developing knowledge-intensive applications (Marcus and McDermott, 2011). So, in this research it is employed a Knowledge engineering method for knowledge acquisition, knowledge model building, knowledge representation and prototype development and testing, whereas other suitable methods are also used for knowledge elicitation through discussion with experts which are professional and experienced teachers and survey design to assess the level of students in understanding compiler error messages, the way how students debug their program and teachers reaction in assisting them. Survey design is more effective in assessing the current practices in its natural setting (Best & Kahn, 2003).

This study was used cross sectional survey design with the intention to get the general picture of the current status of the students and teachers in teaching learning process of computer programming courses and on the way how students debug their program and instructor's reaction in assisting them as well. What are the problems, for both instructors and students, in teaching/learning computer programming? What are the common errors novices' encounters in writing C++ program and what kind of programming style and coding conventions are there? In supporting this idea different authors, (Creswell, 2003) suggested that cross-sectional survey is used to gather data at a particular point in time with the intention of describing the nature of existing conditions or identifying standards against which existing conditions can be compared or determined the relationships that exist between specific events. Moreover, the cross sectional survey design is more effective in assessing the current practices in its natural setting.

1.7.3 Study Area

Mizan Teppi University is one of the higher institutions in Ethiopia which is found in the southern regional state of Ethiopia. It is located in both BenjiMaji and Sheka zones. It has two campus branches located in Mizan town which is the zonal town of BenjiMaji and Teppi which is one of wereda in Sheka zone. The University is giving the social science fields in Mizan campus and natural science fields in Teppi campus. Mizan campus is564 KM and Teppi is 614 KM far from the capital city of Ethiopia (Addis Abeba). Most of natural science students are taking computer programming (computational science, informatics and engineering). However, the researcher was focused on the novice programmer in the University which means students who takes fundamental programming course since the high drop rate is observed in these students in the pre survey. This is not only the reason but it is also noticed by different scholars

Winslow (2013) that the negative impact of these basic introductory courses may have harmful consequences in the learners' attitude towards the next programming courses and totally to field of study.

1.7.4 Data sources

In this study, the researcher has employed both primary and secondary data sources to obtain reliable information about teaching learning computer programming. Sources of data are students and teachers. Different literatures are also reviewed for additional required data.

1.7.4.1 Primary data

The researcher has obtained the primary sources of data from the students and experienced teachers in delivering computer programming courses through questionnaire and interview. The researcher has used questionnaire for students and interview for teachers. Those sources were helped the researcher to acquire first-hand information and to draw inferences.

According to (Dreyfus, 1996), there are five stages of programmers namely; novices, advanced beginners, competent, proficient and expert. As noted by (winslow, 2013) that it takes roughly 10 years to turn a novice into an expert programmer. So, the researcher had two different types of students: very novices (year I) and students with some knowledge of programming (year II) at least have taken two programming courses (Fundamentals of programming I & II). The students from year I, at Mizan Teppi University, were enrolled in the second semester of the first curricular year. They are at the initial stage of learning how to program. In year II, the students are more advanced beginner in learning how to program, they were at the University, enrolled in the second year, second semester, and had already studied introductory aspects of the C++ programming language in the previous semester and developed a semester-long project in C++ (second year informatics students). While they are participating in this research, they are also taking a different course on object-oriented programming in java.

1.7.4.2 Secondary data

The secondary sources of data were obtained through different methods. The researcher was used secondary sources of data by using document analysis, books, handouts, forums, lab manuals and resources from the internet for supporting the primary sources of the data, to collect necessary data related to expert system.

1.7.5 Population of the study and sampling procedures

The population of the study included all informatics first year and second year students (computer science, information technology and information system). Teachers who have the exposure in teaching learning computer programming have also been included in this very research. Accordingly, 643 students and 14 teachers have been taken as a population for this study.

Due to the fact that the student population is mostly too large for the researcher to consider, small but carefully chosen samples were used to represent the population. The sample size will reflect the characteristics of the population from which it is drawn.

There are several methods for determining the sample size. In this study has taken a simple formula from Yamane to determine the sample size. The formula from Yamane (as cited in Robert- and Bas, 2010):

$$n = \frac{N}{1+N(e)^2}$$

Where: n, N & e are sample size, population size and the level of precision respectively. This formula assumes a degree of variability (i.e. proportion) of 0.05 and a confidence level of 95%.

Sample size of students, $n = \frac{N}{1+N(e)^2}$

$$n = \frac{643}{1+643(0.05)^2}$$

`

$$n = \frac{643}{2.6075} = 247$$

Next, sample students have been selected from each department using proportional simple random sampling techniques from a list of department students (see table 1.1). It is a sampling technique appropriate to meet the objective of the study sample.

The population non overlap in the study samples the proportional allocation from each selected department. Furthermore, the data obtain from Mizan Teppi University, University's registrar and each of department's purposively selected to this study since they are on the mandate to provide programming courses and it is the scope of the research.

Table 1.1: Samples considered for the current study

Departments	Year I		Year II	
	Total	**Sampled**	**Total**	**sampled**
Computer science	136	52 $\frac{(136*247)}{643}$	98	38
Information technology	121	46	96	37
Information systems	103	40	89	34

1.7.6 Instrument of data collection

In order to acquire the necessary information from participants, three types of data collecting instruments were used such as Questionnaire, Interview and Observation.

Questionnaire

Both closed and open-ended questionnaires were used to collect quantitative and qualitative data from sampled students. This is because questionnaire is convenient to conduct survey and to acquire necessary information from large number of study subject with short period. Furthermore, it makes possible an economy of time and expense and provides a high proportion of usable response (Best & Kahn, 2003).The questionnaire was prepared in English language, because all of the sample students can have the necessary skills to read and understand the concepts that were incorporated.

The questionnaire has two parts. The first part of the questionnaire describes the respondents' background information; categories include gender, department, and their batch and area of specialization. The second and the largest part incorporate the whole possible reasons on teaching learning process of computer programming courses and where the conceptual difficulty that students are facing.

The researchers were dispatched and collected the questionnaires through the assigned data collectors who are lab attendants in each departments of the school of informatics. To make the data collection procedure smart and cleared from confusions, the data collectors was properly oriented about the data collection procedures by the principal researcher.

Interview

Semi-structured interview were used to gather in-depth qualitative data from 5 computer programming instructors, who are picked from the three departments according to student's preference and school staff member recommendation, within the University on the overall teaching learning computer programming courses.

Because interview has greatest potential to release more in-depth information, provide opportunity to observe non-verbal behavior of respondents, gave opportunities for clearing up misunderstandings, as well as it can be adjusted to meet many diverse situations (Best & Kahn, 2003). The researcher was conducted the interview to get in depth information and used for data triangulation. The interview is presented to experienced teacher in the school about the whole teaching learning process, reasons for grading low in programming, what must be done to motivate, encourage and uplift student in computer programming courses,

Observation

This data collection instrument was used to see the real situation in teaching learning process, challenges that students were facing while practicing in laboratory especially in error handling. Using this method, the researcher had exposure to see different issues like teachers teaching procedures, students learning procedures, problem experience, program developmental tool and student skill in understanding errors and trying to correct them.

1.7.7 Data collection procedures

To answer the research question raised, the researcher went through a series of data gathering procedures. These procedures helped the researcher to get authentic and relevant data from the sample units. Thus, after having letters of authorization from Jimma University for ethical clearance, the researcher directly went to Mizan Teppi University to have a pilot test of the data gathering instruments. To do so, before administrating the questionnaire, the researcher was taken10% of the respondents, which has been taken in to account 25 of students from Jimma University since the university is very near to the researcher. At the end of all aspects related to pilot test, the researcher went to Mizan Teppi University.

The researcher made an agreement with the concerned data collectors having introduced his objectives and purposes. Then, the final questionnaires were administered to sampled students in the selected University, Mizan Teppi University. The participants were allowed to give their own answers to each item independently and the data collectors closely assisted and supervised them to solve any confusion regarding to the instrument. Finally, the questionnaires were collected and made it ready for data analysis.

In addition, the researcher was conducted a kind of semi structures interview with computer programming teachers in Mizan Teppi University (research area) and Jimma University, since it is one of the senior University in Ethiopia and the place where the research is attending his masters class, in order to have an experience exchange. During the process of interview, the researcher attempted to select free and calm environment to lessen communication barriers that disturb the interviewing process.

1.7.8 Method of data analysis

The primary data collected from the survey questionnaire was analyzed on statistical package of SPSS version 20 for windows in order to address the research questions. The data collected from students through closed ended questionnaire (the quantitative one) were processed and analyzed using several sets of statistical tools. Descriptive analysis was employed to have the presentation of the data in frequency and percent.

The qualitative data were organized according to concepts identified from research questions, transcribed and then analyzed according to their major concepts. The results of the qualitative data are then presented using narration. Moreover, the thematic approach was followed to display the analysis and findings from both quantitative and qualitative data. The themes for the data analyses were derived from the conceptual framework of the study that is grounded in the basic research questions. Analysis of quantitative data displayed first and then in corporate by qualitative data analysis in the form of texts and quotes.

1.7.9 Knowledge Acquisition

Knowledge Acquisition (KA) is the process of acquiring relevant knowledge from domain experts and other sources of information such as books, databases, guidelines, manuals, journal articles, computer files, etc. KA is the process of eliciting, structuring and representing (formalizing) domain knowledge acquired from the different sources. The acquired knowledge can be specific to the problem domain, it can be general or it is meta-knowledge (knowledge about knowledge). Knowledge acquisition is the first step and critical task in the development of knowledge based system (Sagheb, 2009).

The knowledge acquisition process of this study consists of activities such as gathering essential knowledge, analyzing the knowledge, identifying vital concepts and modeling the knowledge using decision trees. In this study, to acquire the needed knowledge, both primary and secondary sources of knowledge are used. Before critical knowledge is gathered from the teachers, a preliminary assessment has been done to investigate where students gets conceptual difficulty in learning computer programming. Primary knowledge gathered from experts in the domain area, the instructors of the University in this context, using semi-structured interview. Due to this, the researcher purposely selected 5 computer programming course instructors as per the recommendation of the school staff members, seniority and course exposure.

1.7.10 Knowledge Modeling

Modeling of domain knowledge implies capturing the static structure of information and knowledge types. Decision trees (DTs) are modeling tools that are used in a variety of different settings to organize and break down clusters of data. Similarly, decision tree have been widely used in practical applications area, due to its interpretability and ease of use. Currently, decision trees are used in many disciplines such as medical diagnosis, cognitive science, law and computer diagnosis. The decision tree was used in the three main types of errors (syntax, logical and run time) domain to understand the dimension of the problem. Each tree starts with a set of errors and ends with solutions

1.7.11 Knowledge Representation

The acquired knowledge from the domain experts has been used to represent by using decision tree modeling in formal language logic. Rule based reasoning mechanism has been employed for the inference engine. In knowledge based system there are many reasoning mechanisms; among that the most commonly used are rule based approach, case based approach or the combination of the two. Case based approaches are designed to work in the way that the basic idea of similar problems having similar solutions (Aamodt& Plaza, 2013). It is a rule based System that solves problems by remembering past situations and reusing its solution and lesson learned from it. Case based approach represents situations or domain knowledge in the form of cases and it uses case based reasoning techniques to solve new problems or to handle new situations (Abdulah *et al*., 2014). Rule based reasoning, on the other hand reason from domain knowledge represented in a set of rules.

1.7.12 System Development Approach

Prototyping approach was followed to develop the knowledge based system. Prototyping allows participating users who are students and domain experts for evaluating systems accuracy, performance, effectiveness and efficiency. So that the researcher has developed a prototype of which debug errors occurred in writing computer programming based on the conceptual difficulty that is surveyed from students and the method to correct syntax errors as interviewed programming instructors.

To develop knowledge based systems there are various tools which are available both freely and commercially. Among this SWI Prolog and Lisp are among the most widely used and known frameworks for teaching and academic research purpose (Aamodt& Plaza, 2013). The actual implementation of KBS was based on high level programming languages. However, modern knowledge based system development tools highly depend on their purposes, functionality and some additional features. Based on their purposes, KBS tools are classified as general purpose programming tools such as Java, and framework .NET. On the other hand, there are also specific purpose programming languages such as JRULES, CLIPS, JESS (java expert shell system) (Endris, 2011). In addition programming Language such as C++ provides objects as a mechanism for programmer to control the layout and data structures (Kingston, 2008).

However this prototype is implemented by C++, software which novice programmers are practicing their program, in the intention to be easily usable and accessible for them as an extension for the compiler. Even if prolog is open source software and it is the preferred programming language for developing knowledge based system but it lacks a graphical interface and integration with C++ IDE. Another reason is that because it was found that the rules produced by the knowledge engineer (the researcher this time) could easily be represented by C++.

1.7.13 System Evaluation Method

Once the prototype is developed, the functionality and user acceptance of the system should be tested. So that the evaluation processes focus on systems user acceptance of the prototype and the performance of the system. Accordingly, the system is evaluated by user acceptance testing by preparing questionnaire which is adapted from (Pu*et al*, 2011) that used to evaluate the model called ResQue (Recommender

Systems' Quality of user experience) with users' point of view. Then calculate the total user acceptance by using the following formula:

$$\mathrm{AVP} = \sum_{k=1}^{n} \mathrm{SVi}.\frac{\mathrm{NRi}}{\mathrm{TNR}}$$

Source (Aboneh, 2013)

Where AVP is average performance SV scale value and, TNR total number of respondent and NR is number of respondent

Then the result of user acceptance average performance is calculated out 100%.as follow:

$$\left(\mathrm{AVP} = \sum_{k=1}^{n} \mathrm{SVi}.\frac{\mathrm{NRi}}{\mathrm{TNR}}\right) \mathrm{X}\ 100/\mathrm{NS}$$

Where; NS is number of scale.

Source (Aboneh, 2013)

1.8 Operational definitions

Domain Expert: - is a person who expertise in his/her domain area. In addition, an instructor who provides and facilitates teaching learning process in programming courses is a domain expert in his domain.

Compiler: - is a program which converts the high level language into machine language so that the Integrated Developmental Environment (IDE) can understand what have been written in it as machine language or computer language.

Knowledge Based System: - is the collection of relevant knowledge that is stored in the computer and is organized in such a manner that it can be used for inferences, which is the reasoning process of Artificial Intelligence that takes place in the brain of an Artificial Intelligence process

Novices: - they are new programmers or beginners who have no deep knowledge about programming.

Computer Programming: -. It is a process that leads from an original formulation of a computing problem to executable computer programs.

1.9 organization of the study

This study comprises seven chapters. Chapter one discusses background of the study, the problem statement and research questions, the general and the specific objectives of the study, and methodologies that the researcher used to conduct this study.

Chapter two discusses about theoretical and empirical works review that are relevant for this study. In this chapter, the researcher discussions about artificial intelligence, knowledge bases systems, types of knowledge representation techniques, System Performance Evaluation Methods and related works which are relevant for this study.

Chapter three presents the data presentation, analysis and interpretation of the data gathered by different instruments, mainly questionnaire and semi-structured interview. The summary of the quantitative data is presented by the use of Tables that incorporates various statistical tools. Similarly, the qualitative data was organized according to the themes, analyzed and used to strengthen or to elaborate more that of the quantitative one.

Chapter four of this thesis presents the about the knowledge acquisition processes which show how the required knowledge for system is acquired , how the acquired knowledge is modeled so that it would be easy to represent it into the system and knowledge representation techniques.

Chapter five discusses about Design and Implementation. In this chapter the structural design of the system, knowledge base and inference engine as well as the user interface are presented.

Chapter six discusses about implementation and evaluation of the prototype systems. In this chapter the performance of the prototype is evaluated both the performance of the system and the acceptance of the system by the users.

Finally, the researcher dedicated chapter six for conclusion and recommendation. In this chapter, the researcher discussed the evaluation results and based on the result the researcher presents findings and concludes the study by recommending future works.

CHAPTER TWO
Literature Review

2.1 Concepts of Artificial Intelligence

Technology has become crucial in educational development and for the revolution in learning systems (Olapiriyakul, 2012).Technology creates and transforms the learning and teaching processes, which brings new opportunities to the educational system.
One of such technological advancement is an expert system or a knowledge based system (Rajeswari, 2012).

The main examples of the Knowledge Based System (KBS) developed at the early stages of AI include PUFF (1979), MYCIN (1976), CADUCEUS (1984), QMR (1988), and DENDRAL (1960s and 1970s) and WATSON (2016). Pulmonary function analysis (PUFF) was of the oldest KBS in the field of medicine. It was developed for the interpretation of respiratory tests for diagnosis of pulmonary disorders. Patient inhales/exhales through a tube connected to computerized instrument which measures flow rates and air volumes. PUFF accepts this data along with auxiliary data (age, sex, smoking history), and prints diagnosis in English. As for the knowledge base, a knowledge engineer sat down with an expert pulmonary physiologist at the Pacific Medical Center in San Francisco and developed rules (64 in all). A more recent version of PUFF had about 400 rules.

MYCIN, a precursor to PUFF, was developed for the identification of bacteria in blood and urine samples and prescription of antibiotics 1976. It uses IF-THEN rules (with certainty factors) to represent knowledge. It also interacts with a physician to acquire clinical data. The system asks questions based on current hypothesis and known data and reasons backward from its goal of recommending a therapy for a particular patient. It stores approx. 500 IF-THEN rules, and can recognize about 100 causes of bacterial infection. TEIRESIAS serves as a front-end to MYCIN. It was the first program to provide explanations of how conclusions were reached. TEIRESIAS can answer "why" questions by examining its internal tree of sub goals.

2.2 Knowledge Based Systems

According to Kesarwani & Misra (2013), a knowledge base is the collection of relevant knowledge that is stored in the computer and is organized in such a manner that it can be used for inferences, which is the reasoning process of Artificial Intelligence that takes place in the brain of an Artificial Intelligence process. It is one of the major family members of the AI group. With the availability of advanced computing facilities and other resources, attention is now turning to more and more demanding tasks, which might require intelligence (Kesarwani & Misra, 2013).

KBS can act as an expert on demand without wasting time, anytime and anywhere. KBS can save money by leveraging expert, allowing users to function at a higher level and promoting consistency. In fact, a KBS is a computer based system, which uses and generates knowledge from data, information and knowledge (Sajja & Akerkar, 2010).

Rajeswari (2012) mentioned that these systems are capable of understanding the information under process and can take decision based on the residing information/knowledge in the system, whereas the traditional computer systems do not know or understand the data/information they process. The KBS consists of a Knowledge Base and a search program called Inference Engine (IE). The IE is a software program, which infers the knowledge available in the knowledge base. The knowledge base can be used as a repository of knowledge in various forms. As an expert's power lies in his explanation and reasoning capabilities, the expert system's credibility also depends on the Explanation and Reasoning of the decision made/suggested by the system.

2.3 Knowledge Based Systems Development

Mostly knowledge engineering, the process of building an expert system, involves some basic steps. The main phases of a knowledge based system development processes are planning, knowledge acquisition, knowledge representation and evaluation (Sajja & Akerkar, 2010).The knowledge of the expert(s) is stored in his mind in a very abstract way. Also every expert may not be familiar with knowledge-based systems terminology and the way to develop an intelligent system. The Knowledge Engineer (KE) is responsible person to acquire, transfer and represent the experts' knowledge in the form of computer system (Sajja & Akerkar, 2010).

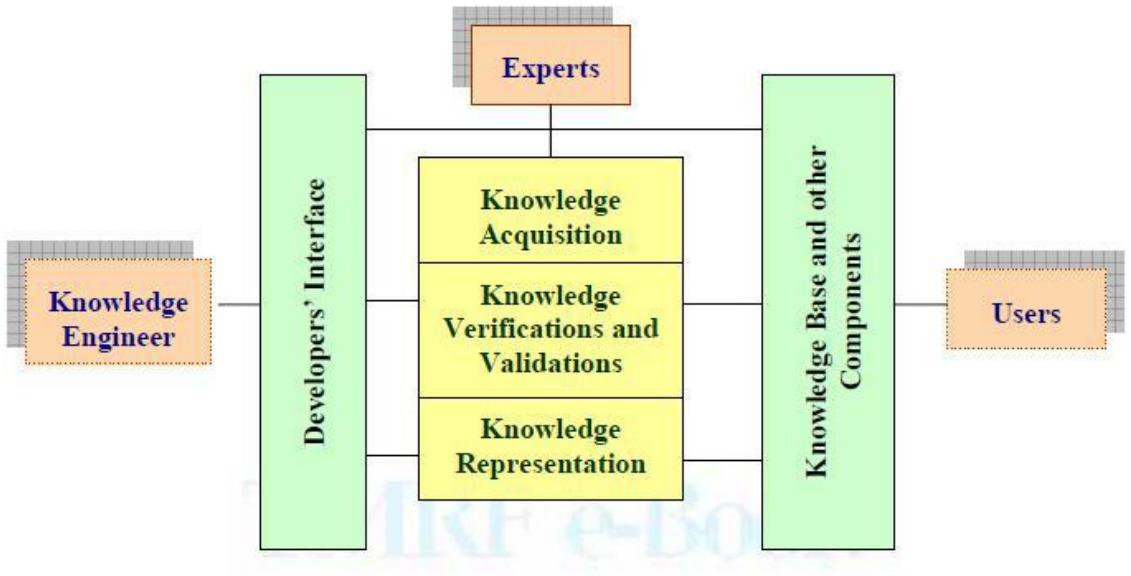


Figure 2.1: Development of a Knowledge-Based System (Sajja & Akerkar, 2010)

2.3.1 Knowledge Acquisition

The knowledge acquisition process incorporates typical fact finding methods like interviews, questionnaires, record reviews and observation to acquire facts and explicit knowledge. However, these methods are not much more effective to extract tacit knowledge which is stored in the subconscious mind of experts and reflected in the mental models, insights, values, and actions of the experts. For this, techniques like concept sorting, concept mapping, and protocol analysis are being used (Sajja & Akerkar, 2010).

The acquired knowledge should be immediately documented in a knowledge representation scheme. At this initial stage, the selected knowledge representation strategy might not be permanent. However, documented knowledge will lead the knowledge engineer/ development to better understanding of the system and provides guidelines to proceed further. Rules, frames, scripts and semantic network are the typical examples of the knowledge representation scheme. It is the responsibility of the knowledge engineer to select an appropriate knowledge presentation scheme that is natural, efficient, transparent, and developer friendly. One may think for hybrid knowledge representation strategies like rules within the frames in slots like “on need” and “on request”; semantic network of default frames etc (Rajeswari, 2012).

2.3.2 Knowledge Modeling

Several key contributions made during the 1980s, including Allen Newell’s notions of knowledge level, William Clancey’s critical analyses and the broader wave of second-generation ES research, have shaped our current perception of the knowledge acquisition problem (Rajeswari, 2012). Central to the current perception is the knowledge model, which views knowledge acquisition as the construction of a model of problem-solving behavior, that is, a model in terms of knowledge instead of representations. The concept of knowledge-level modeling has matured considerably. The practical knowledge level models incorporated in today’s methodologies do not simply reflect the knowledge content of a system; they also make explicit the structures within which the knowledge operates in solving various classes of problems. This enables the reuse of models across applications.

Decision tree

According to (Rajeswari, 2012), decision trees (DTs) are modeling tools that are used in a variety of different settings to organize and break down clusters of data. Similarly, decision tree have been widely used in practical applications area, due to its interpretability and ease of use. Currently, decision trees are used in many disciplines such as medical diagnosis, cognitive science, law and computer diagnosis. Decision tree structures are the bases for the development of prototype knowledge based system.

2.3.3 Knowledge Representation

To build the knowledge base we have the problem of how to represent it. Knowledge representation concerns the mismatch between human and computer ‘memory’. We call these representations, knowledge bases, and the operations on these knowledge bases, inference engine.

A knowledge representation (KR) is an idea to enable an individual to determine consequences by thinking rather than acting, i.e., by reasoning about the world rather than taking action in it. The knowledge acquired from experts or induced from a set of data must be represented in a format that is both understandable by humans and executable on computers. Good Knowledge Representation Languages should be Expressive, Concise, Unambiguous, and Independent of context, Efficient and effective (Kesarwani & Misra, 2013).

Knowledge Representation methods all have advantages and limitations. Production rules are popular in the design of the first-generation expert system. The object-oriented method has become very popular in recent years. Predicate logic provides a theoretical foundation for rule based inferences. To navigate the problem associated with single knowledge representation technique the integrated knowledge representation came into the picture.

Sometimes, no single knowledge representation method is by itself ideally suited for all tasks. When several sources of knowledge are used simultaneously, the goal of uniformity may have to be sacrificed in favor of exploiting the benefits of multiple knowledge representations, each tailored to a different subtask. The necessity of translating between knowledge representations becomes a problem in these cases. Nevertheless, some recent expert system shells use two or more knowledge representation Schemes, e.g., the CORVID, KRYPTON, MANTRA, FRORL system (Kesarwani & Misra, 2013).

2.3.3.1 Frames based Representation

A frame is a node with additional structure that facilitates differentiated relationships between objects and properties of objects. Sometimes it is called as “slot-and-filler” representation. Frames overcome the limitation of semantic network that differentiates relationships and properties of objects. Each frame represents a class or an instance (an element of a class). Frames are application of object-oriented programming for expert systems.

The concept of a frame is defined by a collection of slots. Each slot describes a particular attribute or operation of the frame. Slots are used to store values. A slot may contain, a default value or a pointer to another frame, a set of rules or procedure by which the slot value is obtained (Sharma & Kelkar, 2012).

2.3.3.2 Semantic Networks

Semantic networks are an alternative to predicate logic as a form of knowledge representation. The knowledge can be stored in the form of a graph, with nodes representing objects in the world, and arcs representing relationships between those objects. Semantic network also called as Associative Network.

Semantic representation consists of 4 parts. Part one is Lexical. It tells which symbols are allowed in the representation's vocabulary. Nodes denote objects, links denote relations between objects, and link-labels denote particular relations. The second part is Structural that describes constraints on how the symbols can be arranged. Nodes are connected to each other by links. The third is Procedural which specifies the access procedures (to create, modify, answer questions).Procedures are constructor procedure, reader procedure, writer procedure and erasure procedure. The last part is Semantic that establishes the way of associating the meaning. Nodes and links denote application specific entities.

2.3.3.3Case-Based Representation

Case-Based Representation is a computer technique which combines the knowledge based support philosophy with a simulation of human reasoning when past experience is used, i.e. mentally searching for similar situations happened in the past and reusing the experience gained in those situations (Kesarwani&Misra, 2013). The concept of case based reasoning is founded on the idea of using explicit, documented experiences to solve new problems. The decision maker uses previous, explicit experiences, called cases, to help him solve a present problem. He retrieves the appropriate cases from a larger set of cases. The similarities between a present problem and the retrieved case are the basis for the latter's selection (Rajeswari, 2012).

2.3.3.4 Rule Based Representation

Rule based reasoning is a system whose knowledge representation in a set of rules and facts. Symbolic rules are one of the most popular knowledge representation and reasoning methods. This popularity is mainly due their naturalness, which facilitates comprehension of the represented knowledge. The basic forms of a rule, if<condition> then<conclusion> where <condition> represents premises, and <conclusion> represents associated action for the premises. The condition of the rules is connected between each other with logical connectives such as, AND, OR, NOT, etc., thus forming a logical function. When sufficient conditions of a rule are satisfied, then the conclusion is derived and the rule is said to be fired.

Rules based reasoning was dominantly applied to represent general knowledge. Rule based expert systems have a significant role in many different domain areas such as medical diagnosis, electronic troubleshooting and data interpretations. A typical rule based system consists of a list of rules, a cluster of facts and an interpreter (prentzas & Hatzilygeroudis, 2007).

Rules in the system represent possible actions to take when specified conditions hold items in the working memory. The conditions are usually patterns that must match with the items in the working memory. In forward chaining, actions are usually involved adding or deleting items from the working memory. Interpreter of the inference engine controls the application of the rules, given the working memory. The system first checks to find all the rules whose condition holds true (Rajeswari, 2012).

Rule Based Reasoning Techniques

It is mentioned as there are two main inference methods in rule based reasoning mechanism. These are backward chaining and forward chaining. The former is guided by the goals (conclusions), whereas the latter one is guided by the given facts (Freeman-Hargis, 2014).

Forward chaining

During forward chaining, the inference engines first predetermine the criterion and the next steps are to add the criterion one at a time, until the entire chain has been trained. With data driven control, facts in the system are represented in a working memory which is continually updated. Rules in the system represent possible actions to take when specified conditions hold items in the working memory. The conditions are usually patterns that must match with the items in the working memory. In forward chaining, actions are usually involves adding or deleting items from the working memory. Interpreter of the inference engine controls the application of the rules, given the working memory. The system will first checks to find all the rules whose condition holds true (Nalepa, 2015). Both data driven and goal driven chaining method follows the same procedures. However, the difference lies on the inference process. The system keeps track of the current state of problem solution and looks for rules. This cycle will be repeated until no rules fire or the specified goal state is satisfied (Rajeswari, 2012).

Backward chaining

This strategy focuses its effort by only considering rules that are applicable to the particular goal. It is similar with forward chaining the difference is it receives the problem description as a set of conclusions instead of conditions and tries to find the premises that cause the conclusion. Given a goal state and then the system try to prove if the goal matches with the initial facts. When a match is found goal is succeeded. But, if it doesn't then the inference engine start to check the next rules whose conclusions (previously referred to as actions) match with the given fact. Note that a backward chaining system does not need to update a working memory instead it keeps track of what goal is needed to prove its main hypothesis. Goal driven control is commonly known as top-down or backward chaining (Nalepa, 2015).

2.3.4 Knowledge based system development tools

In the 1980s and early 1990s, when commercial interest in knowledge based system was reach at its peak, approximately there are more than 200 commercially available KBS tools (Sajja & Akerkar, 2010). Many are still available but no longer described as KBS tools for marketing reasons. A knowledge based system tool is a set of computer software that manipulates programs and other information in order to design and assist the development of knowledge based systems (Kesarwani & Misra, 2013). The actual implementation of KBS was based on high level programming languages. However, modern knowledge based system development tools highly depend on their purposes, functionality and some additional features. Based on their purposes, KBS tools are classified as general purpose programming tools such as Java, and framework 1.NET. On the other hand, there are also specific purpose programming languages such as JRULES, CLIPS, JESS (java expert shell system) (Endris, 2011). In addition programming Language such as C++ provides objects as a mechanism for programmer to control the layout and data structures (Kingston, 2008).

There are many knowledge based system tools. According to Kingston (2008) different author classified KBS development tools based on their functionality. The simplistic nature and additional feature it provides is used as parameters to select KBS development tools. Expert systems are typically written in special programming languages. The use of languages like LISP and PROLOG in the development of an expert system simplifies the coding process. The major advantage of these languages, as compared to conventional programming languages, is the simplicity of the addition, elimination, or substitution of new rules and memory management capabilities.

2.3.5 Methods of Evaluation

Knowledge based systems evaluation method can be split into Verification, validation, assessment of human factors and assessment of correctness. These evaluation methods are discussed as follow (Thomas, 2014):

Verification is an evaluation process that should be implemented during system design and development to answer the question Did we build the system correctly'. Verification can be defined as the process that involves checking for compliance with the system specifications, checking for syntactic and semantic errors in the knowledge based system. Specification assessment includes user interface, explanation facility, real time performance and security provisions specified in the system design. To verify a knowledge based system, it is possible to use either a program proof or a test strategy.

The program proof confirms total correctness of the program logic with mathematical methods and the test proof strategy confirms partial correctness of the program with given test cases (s).

Validation: The concept of validation refers to determining the correctness of the system with respect to users' needs. Validation criteria include comparisons with known results (e.g. past cases or solved problem), comparison against expert performance, and comparison against theoretical possibilities. Empirical validation checks whether the results of content remain stable when the system is under full workload. The system test examines the complete system performance in its working environment. Validation tests include user acceptance surveys, direct comparison on random test cases between human expert and that of the system.

Evaluation of human factors is the process of determining the acceptability and usability of the knowledge based system. Usefulness of a system is often measured by examining user satisfaction. User satisfaction measured from different point of views such as content satisfaction, interface satisfaction and institutional objective. Personal aspect such as individuals 'dislike of computer takes into consideration.

Evaluation of explanations is used to evaluate the explanation ability of knowledge based system. An explanation facility must have the ability to accept feedback from the user and provide response for the given feedback. An explanation facility must be able to offer brief description in more than one way. An explanation module should be able to answer a range of questions that a users' wishes to ask and not limited to those questions predicted by developers. An explication module should take into account the user's goals, the problem domain and the previous explanatory dialogue. :

2.4 Related works

Researchers have focused on identification of specific bugs for different reasons: some used their results to create debugging tools, while others to gain insight into computer programming education (Hristova, 2003). Various methods have been used including surveys, interviews, talk aloud exercises, observing students while they solve problems, and hand analysis of program assignments.

A more recent study, Lewis and Gregg (2016) discussed the benefits of introducing certain debugging tools earlier or later in the curriculum. McCartney, (2007) investigated novices debugging strategies and suggested that skills at debugging are distinct from general programming ability which deserves individual attention pedagogically.

Smith and Webb (1992) also did a study called "**Recent Progress in the Development of a Debugging Assistant for Computer Programs**". They made a debugging assistant that provides the users with explicit models of their programs and hence encourage them to find errors for themselves. The transparency debugger called Bradman was created to help novice programmers debug their C programs. The system is an interactive system which builds two models of the user's program one reflecting what the program actually does and the other reflecting what the programmer intended to do. Conflicts between these two models are used by Bradman to find bugs in the program. This way it also provides an active support during the debugging process. They have demonstrated novices appreciate having such information made explicit and that a facility that explains individual statements supports them in their debugging efforts. The limitation of the system is Bradman takes the user's syntactically correct program code as input. This code is parsed and relevant information extracted from each statement. This information is stored in a. tree structure and is called the implementation model. It reflects what the program actually does. At this stage, statements are treated as individual entities and no attempt is made to understand their purpose in relation to other statements.

A research done by Salcedo, Najinar Raysal Marie G (2016) which is entitled as "**novice Assistance in Java Introduction"** is an extension developed for BlueJ on the primary objective of giving a clearer explanation for root cause of a compile error so that it helps new programmers being introduced to Java in their debugging. The researcher has highly argued on the compile errors currently thrown by the compiler don't necessarily point the novice programmers to the right direction. With the help of NAJI (Novice Assistance in Java Introduction), these compile errors are processed to have a more detailed output like background, root cause, and example. The research is done by using the five errors identified in methods and tools for exploring novice compilation behavior to develop an extension of IDE to help novice programmers understand the compiler errors. In this research an object oriented system analysis and design methodology has employed. It is presented as the system is useful but user acceptance and performance test values are not specified (Salcedo, 2016).

A social recommender system that can help different programmers in debugging their program has been done. It is entitled as **HelpMeOut** is supports the debugging of errors by suggesting fixes that peers have applied in the past. It collects examples of code changes that fix errors in a central database. The user feeds the error to a suggestion interface then it queries the database for relevant fixes. In developing the system anova system development methodology is employed. The system is able to suggest useful fixes for 47% of the errors since it is tried to include all types of errors in different programming language C++, java and vb (Bradnt, 2010).

Adil (Automated Debugger in Learning system) is a knowledge-based automated debugger in C language. Stereotyped code and bugs are stored as knowledge base library of plans in the knowledge-base. Adil is able to understand an error-free program and locate, pinpoint, and explain logical errors. It also acts as an IDE by having necessary supporting tools to facilitate the recognition and debugging. Given a syntax error-free program and its specification, this debugger is able to locate, pinpoint and explain logical errors of programs. Knowledge engineering methodology is used (Aljunid*et al*, 2000).

A study made by Lee and Wu, (1999) has findings on improving programming skills of novice programmers by the way they debug. They developed a debugging training which will uncover and correct any misconceptions of the programmers and improve their debugging skills. The model they developed called **DebugIt** covers frequently committed errors in Pascal language. The results showed that the model of supervised debugging was effective in improving novice programmers‘ debugging skills (Lee & Wu, 1999).

Expresso was done in Bryn Mawr College to overcome the problem of cryptic compiler messages (Hristova *et al*, 2003). The primary objective of the research is to help novice java programmers in debugging process. The approach Expresso did is object oriented system analysis design to do a better job of generating error messages and suggesting possible solutions to those errors. It is an educational tool for Java programming. It is mentioned as the tool could enhance the function of the compiler. However, the tool specifically does not eliminate the need for understandable compiler error messages; rather, the tool enhances the functions of a compiler. The intention was to create a helpful interactive tool that would do a better job generating error messages than existing compilers and also provide suggestions on how to fix the code (Hristova *et al*, 2003).

Java Intelligent Tutoring System was a prototype developed to aid in tutoring the language. It focuses on variables, operators, and looping structures. It is a web-based application where you will upload your java program and run your program and returns the output (Sykes & Franek, 2004).

To conclude, several studies have been developed so as to assist students in their debugging skill but they are whether higher levels like java or very low language like pascal and C. there is no debugging assistance system specifically done for C++, which is very popular and working environment to teach fundamentals of programming in many Universities in Ethiopia. So this research is on this very language, C++, to develop a knowledge base debugging assistant system.

CHAPTER THREE
Data Presentation, Analysis and Interpretation

This chapter presents the data analysis and interpretation of the data gathered by different instruments, mainly questionnaire and semi-structured interview on assessing the challenges for both teachers in teaching learning computer programming courses. The data helps as a requirement for the knowledge base system. The summary of the quantitative data is presented by the use of Tables that incorporates various statistical tools. Similarly, the qualitative data was organized according to the themes, analyzed and used to strengthen or to elaborate more that of the quantitative one. Thus the qualitative data is used to support the result obtained from the interpretation of the quantitative data. Data presentation and analysis in this very research is divided into two perspectives: learner's and teacher's perspectives.

3.1 Validity and Reliability of Scale Measures

The validity analysis of the measurement instrument was based on pilot study on 10 % respondents that can be representative of the sample population. The respondents of the pilot study were provided with the original questionnaire and have rated their extent of agreement/disagreement on the statements of the questionnaire. To do so, before administrating the questionnaire, the researcher took 10% of the respondents, which has been taken in to account 25 of students from Jimma University. Furthermore, they have pointed out the shortages of the original data collection instrument by rendering critical suggestions, which are incorporated by revising the survey questionnaire. At the end of all aspects related to pilot test, the researcher went to Mizan Teppi University to distribute the questionnaire for target sample population.

The reliability measurements were calculated on students' side questionnaire for the overall teaching learning process of computer programming of the primary data set by applying internal consistency measurement (Cronbach Alpha). The total average intertermcorrelation/Cronbach alpha coefficient was computed to be(α= 0.836). The value of alpha is close to one (1) indicating a salient level of reliability and well beyond the cutoff point ($\alpha \geq 0.7$) (Leary, 2004).

3.2 Data Presentations and analysis from learner's perspective

The research has conducted by asking 247 students to provide their own perspective of why students fail computer programming courses and drop out of the university. The premise is that there is no better way to find out than to ask the students directly if something has helped and encouraged them to learn and succeed or held them back and discouraged them from learning (Bain, 2004). Actually instructors have also been the participant of the study on strengthening the data on their perspective since teaching learning process is between students and teachers. Accordingly, there are 17 unreturned and lack of full answers questionnaires from the whole sample data. So that the total returned questionnaire is 230 and the researcher presented and analyzed only for the returned questionnaire.

Table 3.1Sex of students

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Male	154	67.0	67.0	67.0
	Female	76	33.0	33.0	100.0
	Total	230	100.0	100.0	

Here is the background information of the student respondents. As depicted in table 3.1 67% of the respondents are male and the remaining 33% are female.

Table 3.2 Department of students

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Computer Science	83	36.0	36.0	36.0
	Information technology	78	34.0	34.0	70.0
	Information system	69	30.0	30.0	100.0
	Total	230	100.0	100.0	

The table above implies that out of the total sample size 36% are computer science students, 34% information technology and 30% information system.

Table 3.3 Student's interest on computer programming

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Yes	159	69.0	69.0	69.0
	No	71	31.0	31.0	100.0
	Total	230	100	100	

As the questionnaire distributed randomly to students implied that students are very interested in computer programming courses. About 69% of the students answered they are interested with the course and they have already joined the school of informatics as per their own interest.

However, student's grade in programming courses on average relative to other courses shows the reverse of their interest, as shown in table 3.4 below.

Table 3.4 Students grade in programming

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Very high	11	5.0	5.0	5.0
	High	36	15.0	15.0	20.0
	Fair	55	24.0	24.0	44.0
	Low	101	44.0	44.0	88.0
	Very low	27	12.0	12.0	100.0
	Total	230	100.0	100.0	

As depicted in the above table 3.4 most of students accounting for 56% scored low and very low grade in comparison only 20% very high and high.

The study also explored the reason for students to score low grade in programming courses, as shown in table 3.5.the main reasons students provided for failing computer programming courses were grouped into seven main categories.

Table 3.5 Reason for students achieving low grade

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	course complexity	157	43.0	43.0	43.0
	teaching methodologies and strategies	51	14.0	14.0	57.0
	student motivation	25	7.0	7.0	64.0
	program development tool	27	7.0	7.0	71.0
	natural language problem	26	7.0	7.0	78.0
	previous experience	70	19.0	19.0	97.0
	way of study	11	3.0	3.0	100.0
	Total	367	100.0	100.0	

Note: as we can see the frequency because the question in questionnaire gives the respondent a chance to select more than one answer.

As presented in table 3.5 the main reason students provided for failing computer programming courses was course complexity with 43%. This is followed by student's previous experiences in any kind of programming (19%), teaching methodologies and strategies (14%), computer program developmental tool and natural language problem (7%), student's motivation (7%) and way of study (3%).

We also raised for respondents the issue of level of student perception for compiler design. Summary of suggestion made by respondents is presented in table 3.6.

Table 3.6 Level of perception for compiler message

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	very easily	23	10.0	10.0	10.0
	Easily	30	13.0	13.0	23.0
	neutral	7	3.0	3.0	26.0
	not easily	89	39.0	39.0	65.0
	even can't understand	81	35.0	35.0	100.0
	Total	230	100.0	100.0	

According to the above table 3.6, Out of the total respondents, 74% (not easily and even can't understand responses altogether) mentioned they can't easily understand what the compiler is notifying while writing computer program. Students understanding of the message forwarded by the compiler while witting computer programming is presented in the above table according to the descriptive analysis of SPSS. It already mentioned by students that programming developmental tool has its own contribution for being stopper in computer programming courses. Stoppers are those students who simply stop and abandon all hope of solving the problem on their own while they notice the program has error and can't fix it with the help of compiler message. Computer program development tools or environments have their own compiler which is built in to the software for notifying students while any syntax and other errors like fatal error. But sometimes it displays many errors for one syntax error which makes student abandon all hope to continue

Table 3.7 Similar frame of mind for teachers in teaching

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly agree	30	13.0	13.0	13.0
	Agree	31	13.0	13.0	26.0
	no idea	23	10.0	10.0	36.0
	Disagree	121	53.0	53.0	89.0
	strongly disagree	25	11.0	11.0	100.0
	Total	230	100.0	100.0	

For the question said "*teacher's teaching frame of mind is always the same.*" Students answered as follow. As the survey implied that 64% of student respondents replied that teacher's mood of teaching is not the same (disagree 53%, strongly disagree11%). Especially teacher's moods in lab are very different. They are very tired in correcting errors where there is large number of student in the lab to follow in every desk, other observation is getting bored and being incomprehensible for students. Here is the presentation.

Table 3.8 Observed teacher's problem in teaching learning computer programming

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	bored	64	28.0	28.0	28.0
	tiresome	92	40.0	40.0	68.0
	Incomprehensible	57	25.0	25.0	93.0
	Not subject knowledgeable	17	7.0	7.0	100.0
	Total	230	100.0	100.0	

As depicted in table 3.8 above, 40% of students have observed there is a tiresome problem on teachers when teaching computer program especially in laboratory class, 28% of students observed getting bored feeling problem on teachers, 25% of students as there are teacher when teacher teach computer programming it is incomprehensible and 7% of students mentioned there are some teachers who are not subject knowledgeable.

It has been seen by this research that there are two kinds of students in teaching learning computer programming. These are stoppers and movers. Stoppers are those students who simply stop and abandon all hopes of solving the problem by their own. Different reasons have discussed with different instructors which is presented later in this very research. While movers are those students who just keep trying, modifying their code and use feedback about errors effectively. According, the frequencies of these students has presented as follows in table 3.9.

Table 3.9 Student's nature in solving problem

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	mover	30	13.0	13.0	13.0
	stopper	200	87.0	87.0	100.0
	Total	230	100.0	100.0	

From table 3.9 one can understand that, there are 87 % stoppers and the remaining 13% are movers.

Table 3.10 further presents some of the difficulty in writing source code using programming language.

Table 3.10 Difficulty for students in writing computer programming

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	source code writing	58	16.0	16.0	16.0
	programming concept understanding	135	37.0	37.0	53.0
	logical design	66	18.0	18.0	71.0
	compiler error correction	105	29.0	29.0	100.0
	Total	364	100.0	100.0	

Note: as we can see the total frequency number is increased since some question in questionnaire gives the respondent a chance to select more than one answer.

The table above shows that, among the given conceptual problems in writing computer program, programming concept understanding comes first with 37%, whereas logical design (18%), source code writing (16%), and compiler error correction (29%).

Having seen the difficulty for student in learning computer programming, the researcher has surveyed every conceptual difficulties of the curriculum. Concepts were gathered from the course outline of fundamentals of computer programming course and then approved by programming instructors. Having approved the concepts the researcher presented the concept to investigate how students got difficulty in these concepts. So, the student respondents were asked to rate every concept of fundamentals of computer programming as high, medium and low. Table 3.11 presents summary of respondent’s suggestion.

Table 3.11 Conceptual difficulties

Concepts	High		Medium		Low		Remark
	Frequency	**Percent**	**Frequency**	**Percent**	**Frequency**	**Percent**	
Variable and data type	62	27%	131	57%	37	16%	**Medium**
Syntax	94	41%	76	33%	60	26%	**High**
Conditional statement	96	42%	87	38%	47	20%	**High**
Switch statement	69	30%	122	53%	39	17%	**Medium**
Loop statement	107	46%	88	38%	35	15%	**High**
Array	110	48%	94	41%	26	11%	**High**
Pointer	122	53%	76	33%	32	14	**High**
Modular programming	126	55%	76	33%	28	12%	**High**
Debugging	85	37%	74	32%	71	31%	**High**
Exception handling	80	35%	104	45%	46	20%	**Medium**
Overall programming	156	68%	46	20%	28	12%	**High**

As depicted in table 3.11 above, syntax, conditional statement, loop statement, array, pointer and modular programming as well as debugging errors are rated as high difficulty. Variable and data type, switch statement and exception handling are medium in their difficulty. Unfortunately there is no low conceptual difficulty rated from the concepts provided for student respondent.

Having asked the student respondents the conceptual difficulty in the curriculum, the researcher has provided the question to ask where student frequently spend their time to study programming courses. Here is the question and its corresponding answer with their frequencies. "*Where do you frequently spend your time to study your programming courses?"*

Table 3.12 Place to study computer programming

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	laboratory	67	25.0	25.0	25.0
	independent study	117	44.0	44.0	69.0
	peer group study	84	31.0	31.0	100.0
	Total	268	100.0	100.0	

Note: as we can see the total frequency number is increased since some question in questionnaire gives the respondent a chance to select more than one answer

The data gathered and summarized in table 3.12 implied that only 25% of students have preferred to study programming in laboratory. But it is believed that programming courses generally contain lots of practical exercises: the issues to be learned do not become concrete for the student until tried in a program (Kordaki, 2010).

Another question presented to student respondent was the place where to find out any kind of help. Here was the question" *Where do you try so as to find answers for every question you have in writing programming*?"Respondents answer presented in table 3.13.

Table 3.13 Place to find answers for questions, if any

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	from teachers	41	15.0	15.0	15.0
	from students	66	24.0	24.0	39.0

from development tool	33	12.0	12.0	51.0
from internet	79	28.0	28.0	79.0
from books	31	11.0	11.0	90.0
no place	29	10.0	10.0	100.0
Total	279	100.0	100.0	

Note: as we can see the total frequency number is increased since some question in questionnaire gives the respondent a chance to select more than one answer.

The cross tabulation analysis between the places where students prefer to have an answer for their question, if any, and the reason for why they prefer it is presented as follows in table 3.14:

Table 3.14 Place to find answers for questions.

Count

		Reason for choosing way to get answer for any question			Total	
		easy to access	easy to use	easy to understand	frequency	percent
Place to find answers for questions, if any	From teachers	14	0	23	37	16
	from students	34	7	22	63	27
	from development tool	13	7	5	25	11
	from internet	38	21	18	77	34
	from books	10	7	11	28	12
	no place	0	0	0	0	0
Total		109	42	79	230	100

The above table depicted that students prefer to use different mechanisms to find the solution for any problem they come across. Accordingly, 16% students prefer teachers for any help (since they are easy to understand and easy to contact), 27% of them preferred their peer students (since they are easy to contact, understand and use), from internet (34%, since it is easy to access, use and understand.

3.3 Analysis and Discussion from Teacher's perspective

In this study, the researcher has also carried out semi-structured interview to 5 instructors who are preferred good by their students, recommended by their staff members, senior and have experience in teaching any computer programming course so as to hear their perspectives on why students fail computer programming courses. The next part of this research report presents the results and discusses the implications of the findings for students and instructors. The researcher has been proposed that being aware of how both students and college of informatics perceive the causes of student failure in academic settings is a necessary step in clinically analyzing the complexity of the problem and in finding workable solutions that could productively lead to helping instructors in teaching and students learn and study computer programming courses.

The study's participants who have experience in teaching computer programming course provided many reasons why some students may fail computer programming courses in the University. Based on the analysis of the answers provided, the reasons for student failure were grouped into four main areas, which were broken into eight categories. The feedback from the face-to-face interview, in-depth discussion with instructors in the school helped in the analysis of the results.

Teacher interviewees perceive that the four main root-cause factors for students failing are (1) student-related factors; (2) course nature; (3) source code developmental tools; and (4) teaching methodologies and strategies (Table 3.15).

Table 3.15. Identified Categories of Root-Cause factors

Major Area	Categories
Student related factors	Previous experience
	Lack of effort
	Lack of motivation

	Personality issues
Course Nature	Course multidisciplinary
Source code developmental tools	Unclear Compiler error message
Teaching methodologies and strategies	Facilities, Materials and Delivery systems
	Lack of student-friendly delivery

Student-related Factors

In the opinion of university school members who responded to the study, the first major area, and largest by far, for failure of students is Student-related Factors. As seen in Table 3.15, under this area there are four categories: (1) previous experiences; (2) Lack of effort; (3) Lack of Motivation or interest; and (4) Personality Issues.

Previous experience

The student-related factor that teacher interviewee mentioned most often was students previous experience in any kind of computer programming even basic computer before. Instructors stated many reasons, including the fact that a significant number of incoming students have poor levels of or a complete lack of academic preparedness for University courses, lack of learning and study skills, and/or lack of the skill of time management and setting priorities. Teacher interviewees cited students' lack of academic preparedness and poor study skills, note-taking skills, reading, and scientific reasoning skills, lack of experience, and more, without directly attributing responsibility. Students lack numerous academic skills, such as critical thinking, and math and science backgrounds even writing skill. All these can be categorized into the main problem of student's previous experience. They have not been adequately prepared for computer programming courses (lack foundational skills such as the ability to think, comprehend the nature of assignment and exams, follow instructions, understanding programming concepts etc.) that interfere with their ability to achieve passing grades. For some reason, many students do not learn these skills throughout grade school and high school even no satisfactory orientation in their very first year, and so when they reach university they are not ready for what it demands.

Teacher interviewees argue that students are not aware of the rigors of their chosen discipline. Many students arrive without knowing how to learn, without having the academic prerequisites, or without having the skill set needed to be successful. A number of students do not realize that university requires a higher level of commitment involving a variety of learning skills, such as deep reading, purposeful study, critical thinking, and browsing different resources like book, internet or even asking for help. As one programming teacher explained "*It is to be seen not only by students who take programming but also by many university students that Students can have difficulty in adjusting their own career expectations. Some students have/aspire to become what they want . . . but they do not realize that it is a very difficult and long road academically . . . some students have not realized this yet.*"

Teacher interviewees saw insufficient academic skills as closely related to lack of time management skills. Target teacher respondents said too many students do not know how to study or learn, do not know how to organize their time and set priorities, do not ask for help from their instructors or advisors, and do not use available resources, such as the library, internet and tutors. They most likely lack critical thinking skills and other higher-level learning skills so necessary in University.

Lack of Effort

The next category of student-related issues was *lack of effort.* Almost all interviewed teachers mentioned as they were disturbed by how many students are satisfied with a grade of C or D instead of working harder to get better grades. They stated that even when they give students opportunities to improve their grades by redoing assignment, lab reports, many students do not bother. Some participants stated that students do not exert enough effort and do not bother to find out, either from the instructor or fellow students, or from book or internet how much work is really needed to pass a given class.

It is said by teacher interviewees that some students expect teachers to excuse multiple missed assignments and absences and to pass just because they attend class. They do not read the material before class and do not complete their assignments. Some students do not care if they fail in programming course. Because they believe they can score well in common courses. A few instructors stated that some students do not value education because they do not have to work to pay for it, or if they fail, they can always repeat the course. And they don't care since their seniors have passed in that way. Students are unable or unwilling to put effort into learning. This could be due to lack of motivation or inadequate preparation to be successful. One teacher respondent explained that many University students do not read to learn: "*In my opinion students fail because they do not put in the effort needed to succeed. They only read in order to answer a question or to pass a test, instead of reading the entire assigned chapters*"

Lack of Motivation or Interest

Lack of Motivation or Interest, engagement, persistence, and "not being active learners" were mentioned frequently in this survey. This category included the following subcategories: Lack of motivation; Lack of engagement; Lack of interest, direction, or focus. Some teacher interviewed thought that failing students have little understanding of how their education relates to their lives. They do not know what they want in life and have no clear goals as to where they are going. Let me use a saying from one teacher interviewed "*If someone has no idea where they are going, it will likely be extremely difficult to get there.*" This was taken from a teacher interviewee. So, most teachers believe in to be done something in student's motivation, interest, and engagement and make them very active learners.

Personality Issues

This category includes Lack of social connection, Lack of support system and network, and Poor self-esteem and self-confidence. One teacher interviewed thought lack of self-confidence was the major reason for failure: here is a direct word from the instructor *"I think most students fail because of a lack in self-confidence. Often the students that I see are bright but make failing grades due to their not believing that they are smart enough to do the work. We try to work through this and if there is some improvement in* *self-confidence, grades improved."*It is also believed by another instructor that Learning is social—no connection to the instructors or the classmates can make a student feel isolated and hence, un-engaged. The general feeling was that if students were "active on campus, and have interactions with the teachers and students outside of the classroom," they would be more likely to succeed in University. One interviewee mentioned teamwork as an important factor in informatics and engineering classes. So teacher interviewees argue on that teachers need to encourage students to increase teamwork sprit. One teacher interviewee concluded that "*for encouraging students self-confidence, we instructors need to familiar students with the programmers communities like programming forums and blogs instead of orienting students only online tutorial websites."*

Nature of the course

Computer Programming is a process that leads from an original formulation of a computing problem to executable computer programs. It is the process of taking an algorithm & encoding it into a notation, a programming language, so that it can be executed by a computer (McCracken, 2014).It is the process of developing and implementing various sets of instructions to enable a computer to do a certain task. Computer programming is linked with Mathematics, Physics, Engineering, Linguistics, Philosophy, Psychology, Economy, Business, and Social Sciences in general (Mulder, 2002).So that one teacher interviewee said *"this nature of the course results in a relatively difficult and very despairing kind of course for students."*However, if supported by suitable teaching strategies and tools it can be mastered by pupils to some extent (Akinola *et al*., 2015).

Interviewee teachers thought that computer programming courses are very heavy in content, but the instructors do not have time to cover the material in depth. They felt that many of the students do not have enough time to absorb the material in the allotted time. One said that by the time students were just starting to understand, he had to move on to the next issue.

Source code developmental tool

Every computer programming has its own developmental environment which enables computer programmer, a person who writes a source code or computer program, to make applicable the algorithm s/he writes. So this developmental environment has its own compiler or interpreter which converts the source code or high level language which is human readable to machine language or computer readable. These compilers or interpreters check any errors in the source code before running. One of the disadvantages of compilers is said one teacher interviewee "*source code developmental environments do not tolerate students for error. Actually the developmental tool notifies the error with its line but not understandable by many novice students. This is because compilers are not always correct they may show 10 errors for one mistake*. *Novice* programmers *often encounter cryptic compiler error messages that are difficult to understand and thus difficult to resolve. Unfortunately, most related disciplines, including compiler technology, have not paid much attention to this important aspect that affects programmers significantly, apparently because it is felt that programmers should adapt to compilers.* "

Teaching methodologies and strategies

This survey includes categories of teacher interviews that do not put fault on students but, instead, on the school and the educational system. The factors in student failures that are not related to students but are related, instead, to the Failures of the teaching methodologies and strategies were mentioned by teacher interviewee. It is well known that students are very new and have no previous experience in any kind of computer programming courses in their previous classes even they may not have experience in using computer. But sometimes staffs forget this big issue. The teacher experts believe that the reason why there is lack of skills on the part of students in solving and analyzing problems in programming is as a result of the poor teaching methodology adapted by programming lecturers. The techniques used by them in the problem representation are not effective. They said that teachers of programming do not employ multiple teaching methods in teaching programming courses. They said that most students lack the understanding of concepts in major topics in programming due to the style/method of teaching. Some teachers only adapt the lecturing method; others project approach, some tutorials, etc. The method either make the course easy to understand or difficult to understand. They believe that most teachers do not consider the background of the students into consideration when teaching programming. They also said that some teachers of programming teach only the theoretical aspect of programming neglecting the practical aspects that will provide the student the necessary skill. In solving this problem, the experts believe that teachers of programming must adapt more than one teaching method to improve their teaching in programming courses to increase the skill and thinking capacity of the student.

Teacher interviewed mentioned that there is a high turnover rate of instructors in the University which can really diminishes the working experience of the University in teaching learning process. This is actually seen by the researcher on carrying out interview with instructors while talking about their seniority. There is a highest experience of 5 years in teaching learning computer programming in the University. So that some of the instructors believe that there is no a kind of instructor assignment for programming course based on specialization or course exposure. This is more the problem of course and exam team in the school. It is said by teacher interviewed that the teaching methodologies and strategies lacks to address students' diverse learning style, how student need to study, encouraging student to use different resources, how to accommodate lab classes, tutoring, where concept have to focus.

Interviewees said that some of their colleagues lack the skills of teaching computer programming. They cited failure to make the subject interesting or relatable, inadequate teaching methods, or failure to inspire. As one teacher explained, *"school members have to take the students from where they are to where they ought to be not from where they think they should be to start, but from where they are. Many students are behind through no fault of their own the school members have to build up student confidence, not tear down student confidence."*

Many of the faculty respondents thought that some faculty members do not put enough effort into engaging the underprepared students in the subject or only help those students who ask for help- no consideration for others. One faculty member was quite passionate about this failure:

The teaching style of some school members was identified as contributing to students' lack of success. Some students had failed because they could not respond to the teaching style, which prevented them from learning, or they had a poor teacher who was unable to effectively communicate the material. As one interviewee put it, in cases like these, "*It is not students who fail, but that faculty fails their students!*" Using different teaching styles and active, problem-solving teaching was offered as the best way to fully engage students. Here are some concepts obtained from teacher interviewees about the factor of teaching methodology and strategies.

Facilities, Materials, and Delivery Systems

Facilities, Materials, and Delivery Systems was mentioned as most often overall root-cause in general and most often root-cause under teaching methodologies and strategies. Lack of tutoring or lack of tutors with the right skills in the subjects was also mentioned as a concern. Attendance has also its own impact. Also, there are those students who stop coming to class but, for some unknown reason, do not withdraw.

A number of faculty members blamed the course delivery format, especially long time for theory class, for failing many students in classes. But the exam is more laboratory oriented. One interviewed teacher said this to strengthen the methodology failure "*we are asking students to answer what we didn't teach them it is like asking a land to produce which is not planted*." The materials sometimes do not get updated. We are still teaching the material that is prepared by other teachers as it is prepare according the knowledge of the author which sometimes may not fit with the teacher. So, instructors need to have the intention to prepare their own material which can go along with their students.

The way some teachers used to deliver the course is also under blamed by the interviewed teachers. It is mentioned that there are some examples which don't exactly show the concept of programming. It is highly believed by teachers that *"examples we used need to show the exact concept we want to explain not for the sake of example* *since we get it from the internet or other materials used. We need to use example which is near to students' eye, the thing they know it well, for easy understanding."*

Another problem observed as mentioned by teachers about in laboratory class is that there observed a feeling of fatigue, getting bored, losing hope about students because they can't easily understand the programming concept. This behavior observed by some teachers especially in laboratory class. Some teachers sometimes forget students come to class to know and the course is complex.

The length of courses was also cited by instructors as a cause for some students. So, teachers need to think over on how to preparing materials which is short and precise and can easily be understood by their students.

Lack of student-friendly delivery

Another problem observed in teaching learning computer programming is lack of student-friendly delivery. One instructor said that "*we are striving to make a change in student, so the course delivery style should be student-friendly and consistent. But some teachers present the course as per their interest and even do not follow the course outline. They amend even the course outline based on their interest; which is very wrong!*"This problem has also observed by students as it is tried to present the students' perspective in table 3.8; it shows as 40% of students have observed there is a tiresome problem on teachers when teaching computer program especially in laboratory class, 28% of students observed getting bored feeling problem on teachers, 25% of students as there are teacher when teacher teach computer programming it is incomprehensible and 7% of students mentioned there are some teachers who are not subject knowledgeable.

3.4 The Need for the Knowledge Based System (Requirements)

As the need assessment conducted revealed that 61% of students in the University have poor academic performance in programming courses, there observed high dropout rate in programming courses. Then to ensure what are the exact challenges for students to score low in computer programming courses should have been addressed, so that the researcher has done an assessment from the perspective of both students and instructors. Based on the finding 69% of students are interested in computer programming courses, but student's grade in programming courses shows the reverse of their interest. Similarly, the study also explored the reason for students to score low grade in programming courses, as shown in table 3.5. Accordingly, the main reasons from the perspective of students were grouped into seven main categories such as course complexity, student's previous experiences in any kind of programming, teaching methodologies and strategies, computer program developmental tool, natural language problem, student's motivation and the way of study. And again from teachers' perspective that the four main root-cause factors for students failing are (1) student-related factors; (2) course nature; (3) source code developmental tools; and (4) teaching methodologies and strategies (Table 3.15).

According to (Winslow, 2013), programming courses generally contain lots of practical exercises: the issues to be learned do not become concrete for the student until start witting a program. So, as tried to show in table 3.6, the level of students' perception for compiler error notification out of the total respondents, 74% (not easily and even can't understand responses altogether) mentioned they can't easily understand what the compiler is notifying while writing computer program. From table 3.9 one can understand that, there are 87 % stoppers and the remaining 13% are movers. In problematic situation stoppers simply stop and abandon all hope of solving the problem on their own, while movers keep trying, modifying their code and use feedback about errors effectively. In being mover the compilers have their own advantage but as shown in table 3.6 level of students' perception for compiler error notification is very low. So, it must be done something to foster the compiler in some extent so that students can easily understand what their problem in the program was. This must be done by providing specific error message to the students. The more specific errors messages from the compiler help students to clarify concepts, misconceptions, or improve their mental models (Salcedo, 2016).

According to Marcus and McDermott (2011) technologies, especially knowledge based system is playing a big role in educational development and for the revolution in learning systems. They bring new opportunities to the educational system. So, this research further go for to what extent the application of knowledge based system support novice programmers in debugging computer program so that knowledge engineering principles followed in the next chapter.

CHAPTER FOUR
Knowledge Acquisition, Modeling and Representation

4.1 Knowledge acquisition

In this study, to acquire the needed knowledge, both primary (tacit knowledge) and secondary sources of knowledge are used. Before critical knowledge is gathered from the instructors, a preliminary assessment has been done to investigate where students get conceptual difficulty in learning computer programming. Accordingly, it is found as students get difficulty in conditional statement, switch statements, array, loop, modular programming and debugging as highest conceptual difficulty out of the course concepts. So, based on this information the researcher gets to sampled instructors to have knowledge for the system. Primary knowledge gathered from experts in the domain area, the instructors of the University in this context, using semi-structured interview. Due to this, the researcher purposely selected 5 senior instructors out of 14 computer programming instructors according to their seniority and course exposure as suggested by school staff members. Accordingly, it was highly advised by instructors to do a solution in debugging process, which rated highly difficult by students, since in debugging it is also possible to teach other concepts which are rated as high difficulty for students like loop, array, module or function. So that the process of knowledge acquisition included some basic activities such as interviewing of domain expert‘s (instructors of computer programming courses), review of relevant sources of information and observing when instructors are correcting errors while students practicing in lab.

According to (Sajja & Akerkar, 2010), mentioned the knowledge acquisition process incorporates typical fact finding methods like interviews, questionnaires, record reviews and observation to acquire facts and explicit knowledge. However, these methods are not much more effective to extract tacit knowledge which is stored in the subconscious mind of experts and reflected in the mental models, insights, values, and actions of the experts. For this, techniques like concept sorting, concept mapping, and protocol analysis are being used (Sajja & Akerkar, 2010). The objective of knowledge acquisition is to gather the required knowledge, interpreting the acquired knowledge, analyzing and validating the knowledge content. Therefore, knowledge acquisition process of this thesis was based on domain expert interviewing, observing and reviewing of related documents, books, lab manuals and guidelines, forums.

Interviewing domain experts

Primary sources of knowledge were collected from human experts in the domain area at Mizan Teppi University computer programming course instructors. To gather the required knowledge semi-structured interview technique was used since one of the main focuses of this research is eliciting relevant knowledge from the domain experts. As already tried to mention five domain experts were selected using purposive sampling technique according to school staff recommendation on seniority and course exposure. The interview with experts covered issues such as how the instructors teaches programming, debugging errors while students faced a problem, what style and coding conventions are there for the concepts.

During the extensive discussion, the researcher acquired the relevant knowledge which was significant to generate the rules. In addition, the domain experts were actively participated throughout the research work and they were consulted to confirm the correctness of the acquired knowledge. During face to face communication, the acquired knowledge from domain experts was recorded manually by using pen and paper sheet. The semi structured interview questions were prepared based on the very discussion with instructors teachers result was discussed below:

The **first** question presented for teacher interviewee was the definition of debugging in writing computer program. Accordingly, teacher interviewees share the same definition for the concept debugging could be defined as the process of finding errors, if any, and then correcting all of them. It is one of the problem happened while writing a computer program and many students are facing. It is also mentioned as an obvious issue as a challenge for many students from informatics department and other field students who take programming courses.

Secondly, the researcher has raised a question for teacher interviewees by leaning on the first question which is based on the definition like if debugging is the process of finding and correct errors, which are known as bugs in programming, how bugs happen in writing computer program. So, teacher interviewees answered as follow. Since programming courses generally contain lots of practical exercises: the issues to be learned do not become concrete for the student until tried in a program. While students trying practical exercises they forget some obvious issues -especially novices students forget some issues even which are even considered as easy concepts like syntax for example. This very statement can be strengthened by Winslow (2013) who noticed that students may know the syntax and semantics of individual statements, but they do not know how to combine these features into valid programs. In general, students need to know the programming style, language rule, language semantic and convention. So, if these issues are violated, then bugs happen in their program.

Thirdly, it was questioned how these bugs could have effect on students practical exercise. When students debug their programs, they may get stuck, which then turns the students into being stopper. Once a student becomes a stopper, s/he starts to fear computer programming courses and can't score a good grade or even drop out from the University. If a student is not able to obtain assistance in debugging their program in a timely and appropriate manner, an excellent educational opportunity turns into a mis-educative one (Dewey, 1997). Instead of practicing proper debugging techniques, the frustrated student learns that programming is difficult, confusing and lonely.

Fourthly, a question "*is these bugs can be debugged?*" was provided to the domain experts. Then accordingly, they answered as there is no bug which can't be corrected. But it needs a skill and knowledge to find the errors and correct them. There is a big difference between novice and expert programmers in debugging computer program. Experts can easily debug their program or other's program as well since they have experience which is developed throughout their life in writing computer program but this skill is not easy for novice programmers.

In the **Fifth** question, what are the general types of errors that are common in C++, the language the novice programmers used? Accordingly, teachers classified error into three common errors; compile time error, run time errors and logical errors. Here, different types of errors have gathered from teacher interviewees which are discussed as follow:

Compile time errors

As acquired from teacher experts, compiler errors are those errors novice students are experiencing every time since they easily forget programming issues like language syntax and semantic. These errors actually can be adapted throughout their practice but sometimes if these errors can't debug early students may give up and stop struggling for the next programming concept since there are stoppers, who don't want to move on while an error experienced. Here are some errors acquired from teacher experts and document reviewed.

Misleading syntax error messages, interpreting syntax errors which point to the wrong line: the compiler in any program development environment wherever there is a syntax error, which is the violation of language rules, it notify error messages. But some students may not understand these messages since they are novice. The development tools sometimes points out to the wrong line which leads novices to confusion.

Missing library: C++ programs are typically created by linking together one or more OBJ files with one or more libraries. A library is a collection of linkable files that were supplied with the compiler. All C++ compilers come with a library of useful functions (or procedures) and classes that you can include in your program. In writing any C++ program, we need to include the libraries we want in the program like Input-output library, mathematical library, and graphical library and so on. Theses libraries run each time you start your compiler. They are called preprocessors. So, many students sometimes forget and miss these library which leads to an error.

Variable not declared: A variable is a symbolic name for a memory location in which data can be stored and subsequently recalled. Variables are used for holding data values so that they can be utilized in various computations in a program. All variables have two important attributes: A type, which is, established when the variable is defined (e.g., integer, float, character). A value is which can be changed by assigning a new value to the variable. Declaring a variable means defining (creating) a variable. You create or define a variable by stating its type, followed by one or more spaces, followed by the variable name and a semicolon. So, variables need to be declared at first before stating to use but novice students forget this issue and try to use a variable which is not declared at this time it will create error.

Variable and data type mismatch: when we create a variable it must be along with its respective data type. Data type is simply the description of a variable. There are different variables such as int, float, char, string, double. Novice students sometimes mismatch variables with their data type. For example to declare a real number they try to declare with int data type but float is correct.

Int and void main function misunderstanding: the actual program begins with a function named *main()*.Every C++ program has a *main()* function. Usually functions are invoked or called by other functions, but *main*() is special. When your program starts, *main*() is called automatically. *main*(), like all functions, must state what kind of value it will return. Many novice student programmers invoking a non-void method without using the return value or they fail to return a value from a non-void method.

Missing semicolon: Since first year students are novice for programming, they may easily forget obvious issues. For example semicolon expected error on a line where there is a statement with a semicolon, but on the previous line the statement is missing its semicolon.

Missing brace and blocks- especially end block: novice programmers miss brace and block. As a rule in programming for every opened block and brace there should be an end block and brace respectively especially in function, conditional statement (nested if), looping statement.

Missing semicolon causes semantic error: a compiler error with a missing semicolon which, when fixed incorrectly leads to a more complex semantic error. The semicolon is actually missing from the line above but the compiler shows a “semicolon expected” error on a line where a semicolon is not needed.

Type mismatch in re-declaration: when novices write a modular program they sometimes mismatch data type of the function while trying to define it. This simply means that when a function is declared with a data type then it must be defined with similar data type. but whenever trying to define with no data type or different data type then this time the compiler sends an error message ”*type mismatch in re declaration*”.

Call to undefined function: this error message would be displayed whenever there is undefined function used in calling or a function defined and called is different in naming.

Linker error: undefined symbol: a linker error which express as there is a kind 4of undefined symbol if a novice is trying to define a function which is not already declared or even different function name in declaration and definition is used.

‘)’ Expected: this error happen because of having extra open brace for function declaration or IF or even loop.

Logical errors

These errors are done while running the program whenever there is a logical wrong. This is the very difficult issue for novice programmers to correct.

Confusion between assignment (=) and equals to operator (==): the incorrect use of an assignment operator (=) when an equality (==) operator is needed. A single equals is actually an assignment operation and the type value of an assignment is based on the type of the left hand side of the statement. The Boolean expression is then corrected by using == instead of =.

Confusion between and (&&) and or (||) operators: novice programmers committed confusion on using logical operators for example and & or operators. They use and operator (&&) when the need to use or (||) operator so that there happen logical error. Sometimes novices use single operator (&) and (|) instead of using (&&) and (||).

Arithmetic expressions with operator precedence problems: in writing a program for arithmetic expression, novice programmers miss the precedence between arithmetic operators so that they do a logical error which results beyond the expected output.

Improper use of modulus: modulus is an arithmetic operator which helps to calculate the remaining number while dividing a number. Novice students use this arithmetic operator in logical programming intension which creates an error.

Infinite loop: This occurs due to lack of update of a loop iterator, the condition statement or inside an if statement in the loop. It identifies that an infinite loop is happening when the program does not seem to end. In looping statement there are there conditions to be satisfied such as initialization, condition and increment. Novice programmers do wrong in conditional statement which results unstoppable output.

Missing input statement: This shows an infinite loop that occurs due to a lack of an input statement inside a validation loop. If the novice programmers miss the input statement which is called initialization then there will be error in the program.

Using improper data type for a variable: novice programmers do a mistake in selecting a data type for a variable. They use int for a variable which is expected the output to be float. This simply means that they use integer data type for real number. Example **int average** or **char name**.

Missing break statement in switch: novice programmers miss a break statement in switch so that unexpected result would be seen in the output.

Wrong condition in if or loop statements: when novice programmers write a program especially which includes if or loop statements, they use a wrong condition that leads the program to display an output which is not expected

Uninitialized variables: when using a loop, variable must be declared and initialized before the condition is checked but novice students trying to run a loop statement with uninitialized variable which is only declared. For example

```
int count;
while (count < 100)
{
cout<< count;
count++;
}
```

This time the compiler compilers well but cannot output any result just silent!

Setting a variable to an uninitialized value: frequently novice programmers believe that variables work like equations - if you assign a variable to equal the result of an operation on several other variables that whenever those variables change, the value of the variable will change. In C++ assignment does not work this way: it's a one shot deal. Once you assign a value to a variable, it's that value until you reassign the values. In the example below, because a & b are not initialized, sum will equal an unknown **random number** from the integer range since a & b are declared as integer, no matter what the user inputs.

```
int a, b;
int sum = a + b;
cout<< "Enter two numbers to add: ";
cin>> a;
cin>> b;
cout<< "The sum is: " << sum;
```

Run time errors

Array index out of bound: this shows the runtime error that results when an index references beyond the end of an array. It points out that one should always look both at the line number in a runtime error and the index listed in an "ArrayIndexOutOfBounds" error. This error is fixed by changing the constant index values which are used to index the array. A general tip is given at the end that usually one indexes arrays with a variable and one should watch that the value of that variable does not become too large.

Unhandled exception (Divide error exception): this very error happens while novice students are trying to divide a number by zero. In this case the compiler will return a run time error which says unhandled exception or divide error exception.

Unhandled exception (General protection exception): this very error happens while novice students are trying to use less than symbol for 'cout' statement if different print statements are intended to use in one cout. Example cout<<"hello"<" world"

Wrong input to output display window: Novice programmers do error in inputting a data to the output display window which is illegal. When they are expected to input an integer value then they sometimes enter another type of data because sine they may do wrong in asking the user while using cout statement.

In the **sixth** question, it was asked the instructor “*What do students do when they get stuck while they debug their program?*”When possible, they turn to their instructors, a lab assistant or perhaps a classmate or senior friend if such human help is readily available. If it is not, they may email someone for help or search their textbook; they may even search the internet.

The **seventh** question was raised about feasibility of human intervention in student help. Of course, it is not feasible to provide human intervention on 24/7 basis. So, the question “*what is the best thing to being there*?” was followed and a kind of knowledge based system which identifies different errors that are experienced by novice student programmers and the help them in their practical exercise to debug their code and helping them to develop debugging skills.

Document review

The researcher has reviewed different additional documents so as to find errors that novice students are experiencing while writing a computer program especially in C++. Ahmadzadeh *et al*, (2005) only look at errors that can be found by compiler. It is mentioned by ahmedzadeh *et al* (2005), all these errors must be faced by student before any other debugging begins. Perkins *et al* (2014) has tried to explore some of lists of the run time errors which experienced by beginner programmers. Oman *et al* (2011) has viewed the semantics of some common compile time errors which is done by novice programmers.

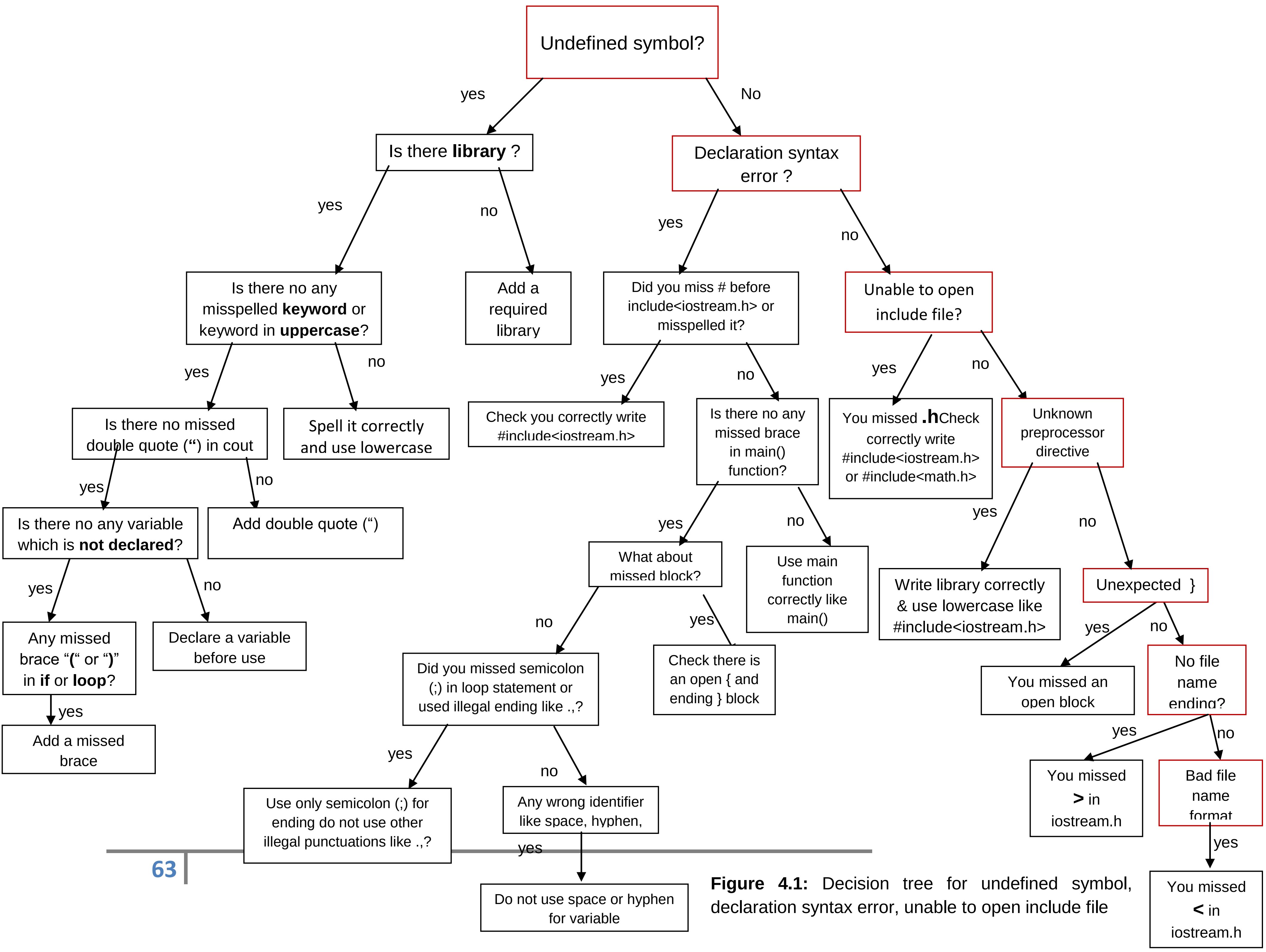
Observation

Knowledge has explored by observing students while practicing program writing in laboratory. Accordingly, the strategies that students use when debugging their programs has acquired. Students have compared by their debugging practices of novice programmers with those of advanced programmers and have found differences in effectiveness, efficiency and techniques used. Students tend to use different debugging strategies based on their familiarity with the program being debugged. Accordingly, some students have observed on focusing on successive lines of code while debugging a program- especially when debugging the work of another programmer. Other students were trying to locate the bug based on program output- especially observed when debugging their own code.

4.2 Conceptual Modeling

Conceptual Modeling of domain knowledge implies capturing the static structure of information and knowledge types. Decision trees (DTs) are modeling tools that are used in a variety of different settings to organize and break down clusters of data. Similarly, decision tree have been widely used in practical applications area, due to its interpretability and ease of use. Currently, decision trees are used in many disciplines such as medical diagnosis, cognitive science, law and computer diagnosis. The decision tree was used in the three main types of errors (syntax, logical and run time) domain to understand the dimension of the problem. Each tree starts with a set of errors and ends with solutions.

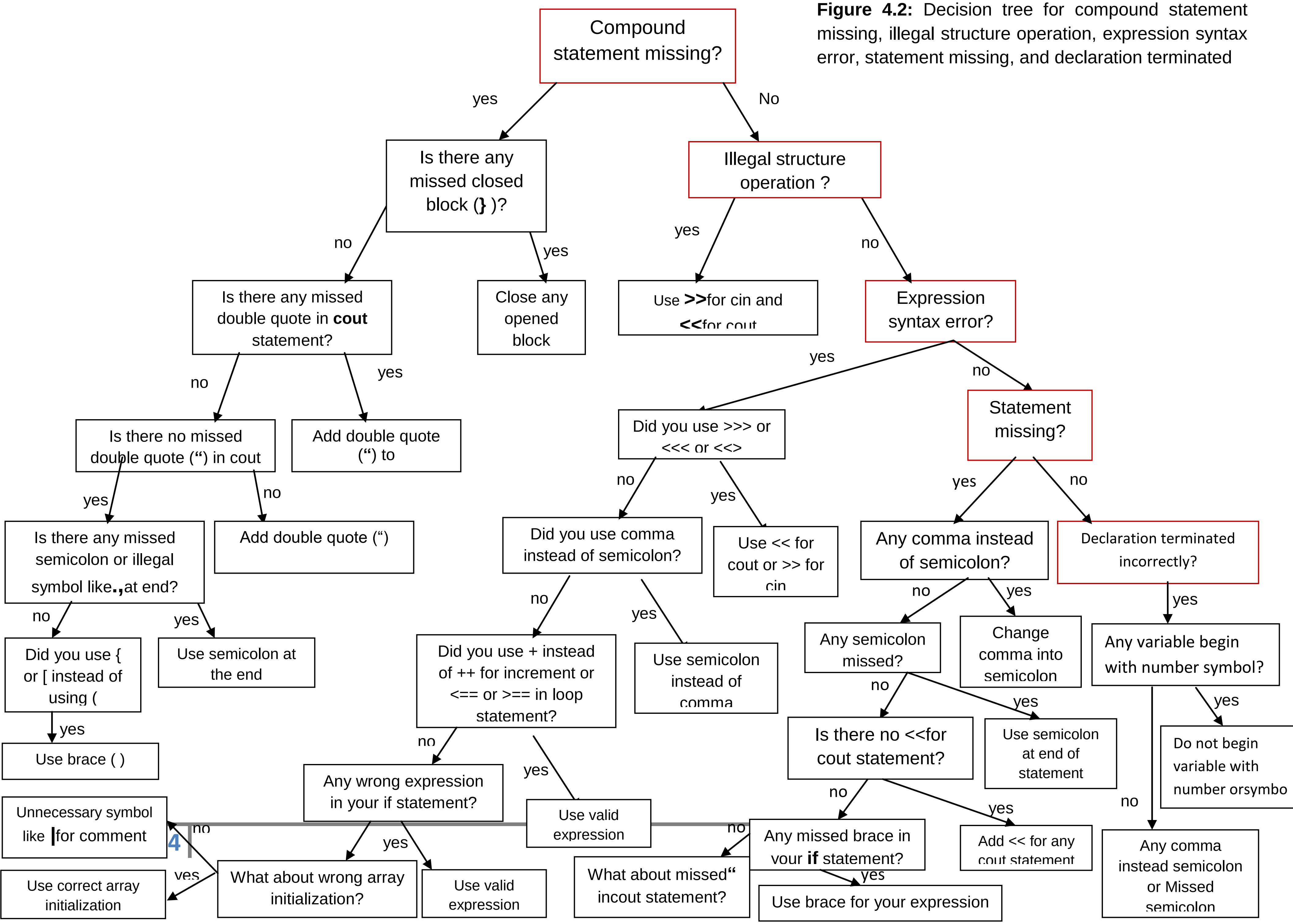
Decision tree structures are the bases for the development of prototype knowledge based system. The prototype follows the same procedures as presented in the decision tree when finding and correcting errors in any program. The system is implemented as defined in the succeeding diagrams. Generally, the tool's input and output requirements are defined in the framework below.



Undefined symbol?
yes
No
Is there library ?
Declaration syntax error ?
yes
no
yes
no
Is there no any misspelled keyword or keyword in uppercase?
Add a required library
Did you miss # before include<iostream.h> or misspelled it?
Unable to open include file?
yes
no
yes
no
yes
no
Is there no missed double quote (“) in cout
Spell it correctly and use lowercase
Check you correctly write #include<iostream.h>
Is there no any missed brace in main() function?
You missed .h Check correctly write #include<iostream.h> or #include<math.h>
Unknown preprocessor directive
yes
no
yes
no
yes
no
Is there no any variable which is not declared?
Add double quote (“)
What about missed block?
Use main function correctly like main()
Write library correctly & use lowercase like #include<iostream.h>
Unexpected }
yes
no
no
yes
yes
no
Any missed brace “(“ or “)” in if or loop?
Declare a variable before use
Did you missed semicolon (;) in loop statement or used illegal ending like .,?
Check there is an open { and ending } block
You missed an open block
No file name ending?
yes
yes
no
yes
no
Add a missed brace
Use only semicolon (;) for ending do not use other illegal punctuations like .,?
Any wrong identifier like space, hyphen,
You missed > in iostream.h
Bad file name format
yes
yes
Do not use space or hyphen for variable
You missed < in iostream.h

Figure 4.1: Decision tree for undefined symbol, declaration syntax error, unable to open include file

Figure 4.2: Decision tree for compound statement missing, illegal structure operation, expression syntax error, statement missing, and declaration terminated


Compound statement missing?
yes
No
Is there any missed closed block (})?
Illegal structure operation ?
no
yes
yes
no
Is there any missed double quote in cout statement?
Close any opened block
Use >>for cin and <<for cout
Expression syntax error?
no
yes
yes
no
Is there no missed double quote (“) in cout
Add double quote (“) to
Did you use >>> or <<< or <<>
Statement missing?
yes
no
no
yes
yes
no
Is there any missed semicolon or illegal symbol like.,at end?
Add double quote (“)
Did you use comma instead of semicolon?
Use << for cout or >> for cin
Any comma instead of semicolon?
Declaration terminated incorrectly?
no
yes
no
yes
no
yes
yes
Did you use { or [instead of using (
Use semicolon at the end
Did you use + instead of ++ for increment or <== or >== in loop statement?
Use semicolon instead of comma
Any semicolon missed?
Change comma into semicolon
Any variable begin with number symbol?
yes
no
yes
no
yes
yes
Use brace ( )
Any wrong expression in your if statement?
Is there no <<for cout statement?
Use semicolon at end of statement
Do not begin variable with number orsymbo
Unnecessary symbol like |for comment
no
yes
Use valid expression
no
yes
no
Any missed brace in your if statement?
Add << for any cout statement
Any comma instead semicolon or Missed semicolon
4
yes
yes
no
yes
Use correct array initialization
What about wrong array initialization?
Use valid expression
What about missed“ incout statement?
Use brace for your expression

4.3 Knowledge representation

The acquired knowledge should be immediately documented in a knowledge representation scheme. To build the knowledge base we have the problem of how to represent it. Knowledge representation concerns the mismatch between human and computer 'memory'. We call these representations, knowledge bases, and the operations on these knowledge bases, inference engine.

A knowledge representation (KR) is an idea to enable an individual to determine consequences by thinking rather than acting, i.e., by reasoning about the world rather than taking action in it. The knowledge acquired from experts or induced from a set of data must be represented in a format that is both understandable by humans and executable on computers. Good Knowledge Representation Languages should be Expressive, Concise, Unambiguous, and Independent of context, Efficient and effective (Kesarwani&Misra, 2013).

Rule based representation

Rule based reasoning mechanism were employed for the inference engine. In knowledge based system there are many reasoning mechanisms; among that the most commonly used are rule based approach, case based approach or the combination of the two. Case based approaches are designed to work in the way that the basic idea of similar problems having similar solutions (Aamodt& Plaza, 2013). It is a rule based System that solves problems by remembering past situations and reusing its solution and lesson learned from it. Case based approach represents situations or domain knowledge in the form of cases and it uses case based reasoning techniques to solve new problems or to handle new situations (Abdulah*et al*., 2014). Rule based reasoning, on the other hand reason from domain knowledge represented in a set of rules. The basic format of a rule is

> *IF <condition> THEN <conclusion>, where <condition> represents premises and <conclusion> represents associated action for the given premises*.

Rule based reasoning is a system whose knowledge representation in a set of rules and facts. Symbolic rules are one of the most popular knowledge representation and reasoning methods. This popularity is mainly due their naturalness, which facilitates comprehension of the represented knowledge. The basic forms of a rule, if<condition> then <conclusion> where <condition> represents premises, and <conclusion> represents associated action for the premises. The condition of the rules is connected between each other with logical connectives such as, AND, OR, NOT, etc., thus forming a logical function. When sufficient conditions of a rule are satisfied, then the conclusion is derived and the rule is said to be fired.

Rules based reasoning was dominantly applied to represent general knowledge. Rule based expert systems have a significant role in many different domain areas such as medical diagnosis, electronic troubleshooting and data interpretations even in teaching concepts. A typical rule based system consists of a list of rules, a cluster of facts and an interpreter (Rajeswari, 2012).

It is mentioned as there are two main inference methods in rule based reasoning mechanism. These are backward chaining and forward chaining. The former is guided by the goals (conclusions), whereas the latter one is guided by the given facts (Freeman-Hargis, 2014). During forward chaining, the inference engines first predetermine the criterion and the next steps are to add the criterion one at a time, until the entire chain has been trained. With data driven control, facts in the system are represented in a working memory which is continually updated. Rules in the system represent possible actions to take when specified conditions hold items in the working memory. The conditions are usually patterns that must match with the items in the working memory. In forward chaining, actions are usually involves adding or deleting items from the working memory. Interpreter of the inference engine controls the application of the rules, given the working memory. The system will first checks to find all the rules whose condition holds true (Nalepa, 2015).

The backward chaining focuses its effort by only considering rules that are applicable to the particular goal. It is similar with forward chaining the difference is it receives the problem description as a set of conclusions instead of conditions and tries to find the premises that cause the conclusion. Given a goal state and then the system try to prove if the goal matches with the initial facts. When a match is found goal is succeeded. But, if it doesn't then the inference engine start to check the next rules whose conclusions (previously referred to as actions) match with the given fact. Note that a backward chaining system does not need to update a working memory instead it keeps track of what goal is needed to prove its main hypothesis. Goal driven control is commonly known as top-down or backward chaining (Nalepa, 2015).

According to (Nalepa, 2015), both forward chaining and backward chaining have similar function. But, the difference occurs due to the data structure of the knowledge based system. So, as noted by (Nalepa, 2015) the backward chaining is more efficient if there is particular goal and avoid drawing a conclusion from irrelevant facts, the developed prototype infers the rules by backward chaining and provides appropriate recommendations as per the users query.

CHAPTER FIVE
Design and Implementation of the prototype

A knowledge based system tool is a set of computer software that manipulates programs and other information in order to design and assist the development of knowledge based systems (Kesarwani & Misra, 2013). In the 1980s and early 1990s, when commercial interest in knowledge based system was reach at its peak, approximately there are more than 200 commercially available KBS tools (Sajja & Akerkar, 2010). Many are still available but no longer described as KBS tools for marketing reasons. The actual implementation of KBS was based on high level programming languages. However, modern knowledge based system development tools highly depend on their purposes, functionality and some additional features. Based on their purposes, KBS tools are classified as general purpose programming tools such as Java, and framework .NET. In addition programming Language such as C++ provides objects as a mechanism for programmer to control the layout and data structures (Kingston, 2008).

There are many knowledge based system tools. According to Kingston (2008) different author classified KBS development tools based on their functionality. The simplistic nature and additional feature it provides is used as parameters to select KBS development tools. Expert systems are typically written in special programming languages. The use of languages like LISP and PROLOG in the development of an expert system simplifies the coding process. The major advantage of these languages, as compared to conventional programming languages, is the simplicity of the addition, elimination, or substitution of new rules and memory management capabilities

This knowledge based novice assistance, is a tool that aids novice programmers in debugging computer programs source code. The user compiles their code in turbo and if there’s a compile error, the KB-DAS system first determines the compile error. The second step is it provides information on the error and suggests the fixes.

Without the tool, the error messages by the compiler are not helpful and often lead to confusion for novice programmers. Too often, the compile error messages are cryptic, long or hard to understand. These don't necessarily point the students in the right direction needed to fix the code. New students of the language have a hard time identifying the errors, let alone applying fixes.

With the tool, the errors are described clearly, real causes are pointed out, and better error messages are generated. It's useful during the early phase of learning C++ programming as they become more proficient and knowledgeable with the language. Once the user compiles the code and a compile error results, the tool then assesses the line which caused the error and processes the code. It scans the code to check the syntax, points out the actual error, and suggests the fix for it. It also gives examples for the students to have better understanding.

Users gain self-confidence and experience in debugging with the assistance of this tool. They are able to save time and improve their program comprehension skills.

5.1 How KB-DAS interacting with users

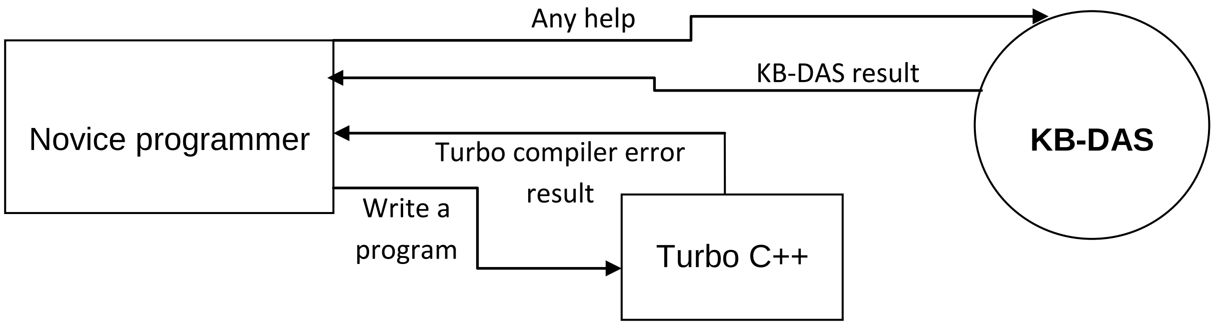


Figure 5.1: structural design of KB-DAS

Figure 5.1 depicts how the prototype works during helping novices students debugging computer program written in C++ language. Accordingly, the novice programmer first write his or her program in turbo C++ developmental tool then if there is any error happen to the program and error message is arise then he or she go to the knowledge base debugging assistance system by having that error message for help then the system asks the programmer some question to suggest a solution.

5.2 Architecture for KB-DAS

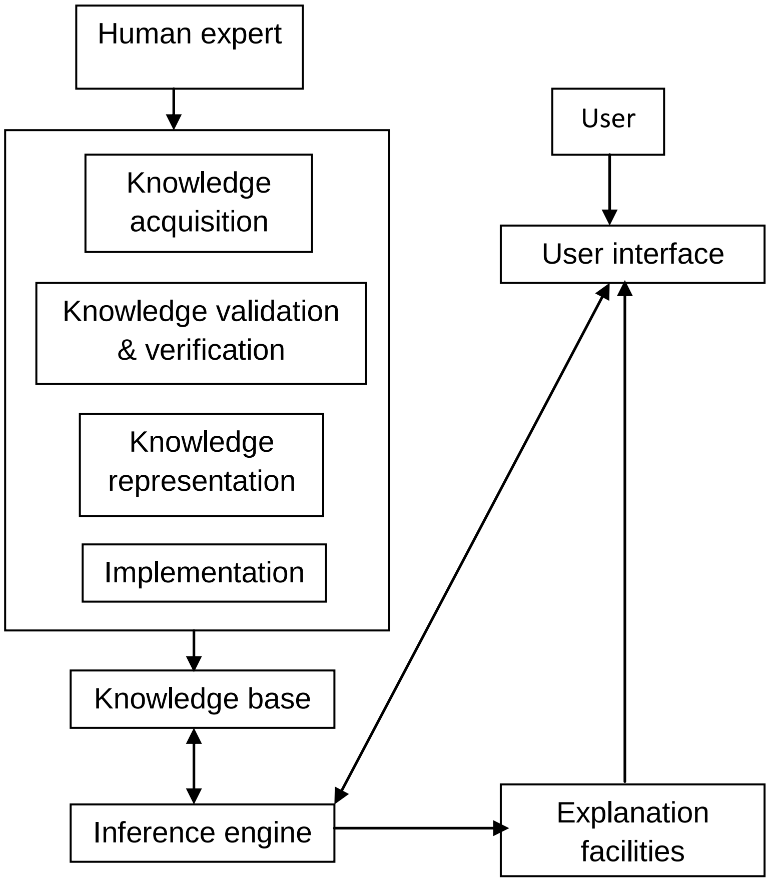


Figure 5.2: architecture of KB-DAS

As it could be observed in the figure 5.2, there is no a learning component for the knowledge base debugging assistance system which is one of the limitation of this research.

5.3 The knowledge base

The knowledge base incorporates the relevant knowledge that was acquired from the domain experts. The knowledge base stores all relevant knowledge, fact, rules, and relationships used by the KBS. The knowledge base of the prototype contains the domain knowledge which is used to identify the types of error and solutions in debugging computer program for novice programmer.

The fact base component of knowledge based system includes basic facts of different cases that are handled during problem solving. The number of facts depends on the number of rules incorporated into the knowledge base. Functionally, the facts in the fact base are used to compare against the condition part of rules. In other words, if the given facts satisfy all the conditions which proved to be true, then the inference engine draw a conclusion. This is based on the pattern matching between the facts in fact base and their respective rules in the knowledge base.

5.4 The inference engine

Two most general types of inference are: forward chaining and backward chaining. Furthermore, combinations of the two types can be applied. The most typical strategy is to use forward chaining as a general control strategy, while at some stages, if detailed goals are to be inferred, backward chaining is employed. Forward chaining is guided by the goals or conclusions, whereas the backward chaining is guided by the given facts. The inference engine simulates the domain expert reasoning process in debugging any computer program errors written in turbo C++. It works from the facts in the working memory (fact base) and stored knowledge in the knowledge base to fire the rule. It achieves the goal by searching through knowledge base to find rules whose premises match with the given facts in working memory. The searching process continues until the inference engine unable to match any premise with the facts in the working memory.

As the result, the prototype system uses backward chaining reasoning mechanism. During the reasoning process, the inference engine start from the consequence (from the problem or error occurred) and checks the reasons of the occurrence of this error message to provide suggestions for the problem. If certain antecedents (facts) are evaluated as true, then it logically follows the consequent are proved, and then the problem type, cases and solutions for the problems are provided. As the conceptual model indicated in the decision tree of figure 4.2, during the debugging any programming errors the system first asks the cases of the error. Next the general practitioner tries to prove whether these causes are match with the causes in the knowledge base or not. Then the system provides suggestion to the novice programmer depending on the actual causes feed to the system. The inference engine of the rule based system follows similar procedures like the general practitioner(s). The inference engine sequentially searches each rules then draw the conclusion for the errors. The rules that used to debug computer program errors in turbo C++ programming language are represented as indicated below.

Rule 1: undefined symbol, if

> There is no library **or**
>
> There is misspelled keyword or keyword in uppercase **or**
>
> There is any missed double quote in cout statement **or**
>
> There is any used variable which is not declared first **or**
>
> There is any missed brace in if or loop statements or
>
> ‘endl’ is not correctly spelled **or**

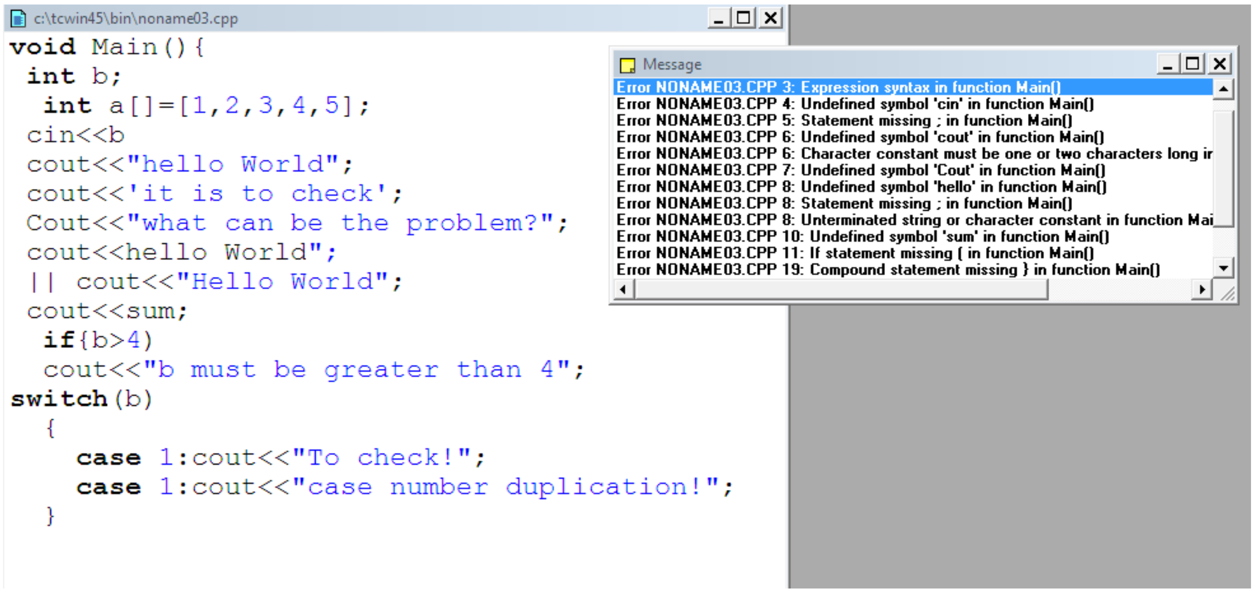
Rule 2: declaration syntax error, if

> There is missed # in libraries and misspelled **or**
>
> There is missed brace in main function **or**
>
> There is any missed block for main function **or**
>
> There is any missed semicolon in loop or illegal punctuation for statement ending like comma, colon or period **or**
>
> There is any wrong variable declaration like space or hyphen.

Others remaining production rules are attached with the document in the appendix [*appendix IV*]

5.5 The user interface

The acceptability of a KBS depends on the quality of the user interface. The user interface is used as the means of interaction between a user and the knowledge base system. For this Knowledge base debugging assistance system, novice users are able to interact with the system through a yes or no response only. Based on the user‘s response the system draws a conclusion for each rule in the knowledge base. The user’s response helps the knowledge base system as premises for drawing to conclusion. The systems conclusion displayed in the user interface window. The figure below shows when the novice programmer write a computer program then tries to debug it the compiler notifies the following errors. The sample code is one of the test cases with 12 compiler errors [*appendix vi*].

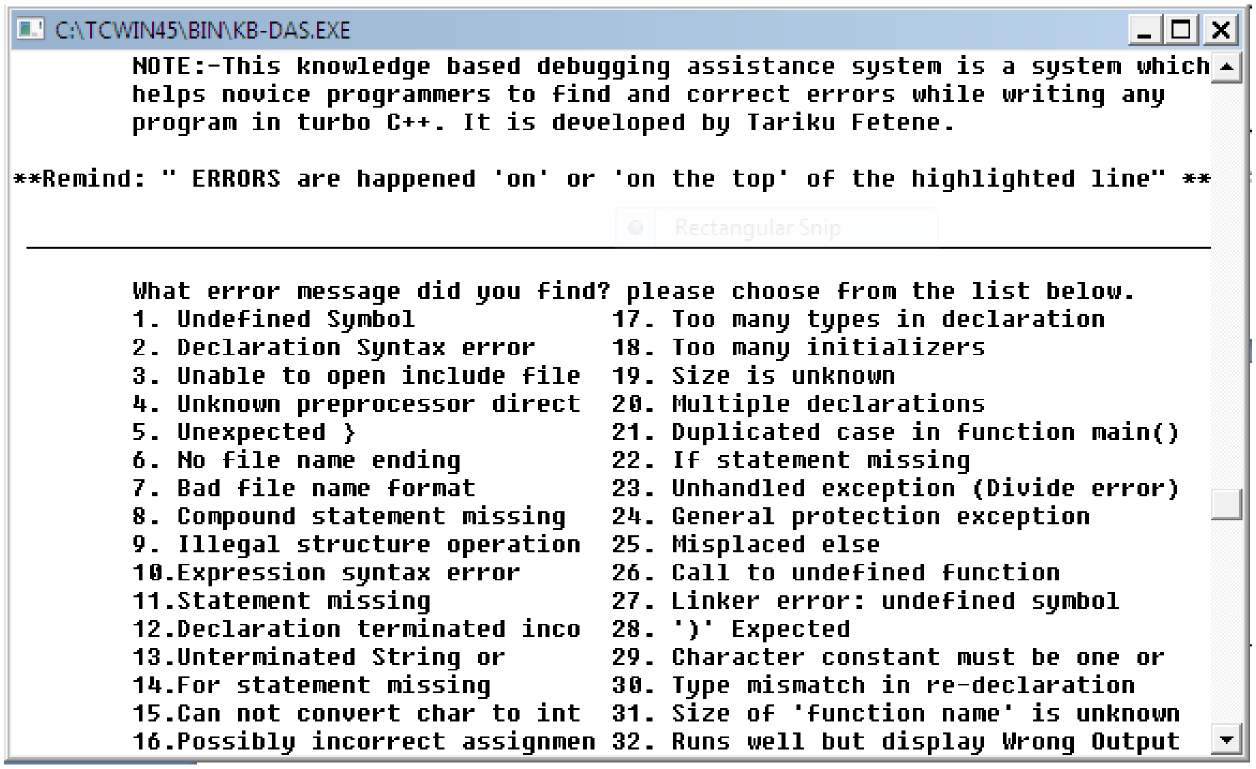


```
void Main(){
 int b;
  int a[]=[1,2,3,4,5];
 cin<<b
 cout<<"hello World";
 cout<<'it is to check';
 Cout<<"what can be the problem?";
 cout<<hello World";
 || cout<<"Hello World";
 cout<<sum;
  if{b>4)
  cout<<"b must be greater than 4";
switch(b)
  {
    case 1:cout<<"To check!";
    case 1:cout<<"case number duplication!";
  }
```

```
Error NONAME03.CPP 3: Expression syntax in function Main()
Error NONAME03.CPP 4: Undefined symbol 'cin' in function Main()
Error NONAME03.CPP 5: Statement missing ; in function Main()
Error NONAME03.CPP 6: Undefined symbol 'cout' in function Main()
Error NONAME03.CPP 6: Character constant must be one or two characters long ir
Error NONAME03.CPP 7: Undefined symbol 'Cout' in function Main()
Error NONAME03.CPP 8: Undefined symbol 'hello' in function Main()
Error NONAME03.CPP 8: Statement missing ; in function Main()
Error NONAME03.CPP 8: Unterminated string or character constant in function Mai
Error NONAME03.CPP 10: Undefined symbol 'sum' in function Main()
Error NONAME03.CPP 11: If statement missing ( in function Main()
Error NONAME03.CPP 19: Compound statement missing } in function Main()
```

Figure 5.3: error message by compiler when the user tries to debug own program

So, here after the novice programmer starts asking a help from a system called KB-DAS, which is knowledge based system developed for this purpose. The figure below shows the welcome window and it describes the functions of KBS.

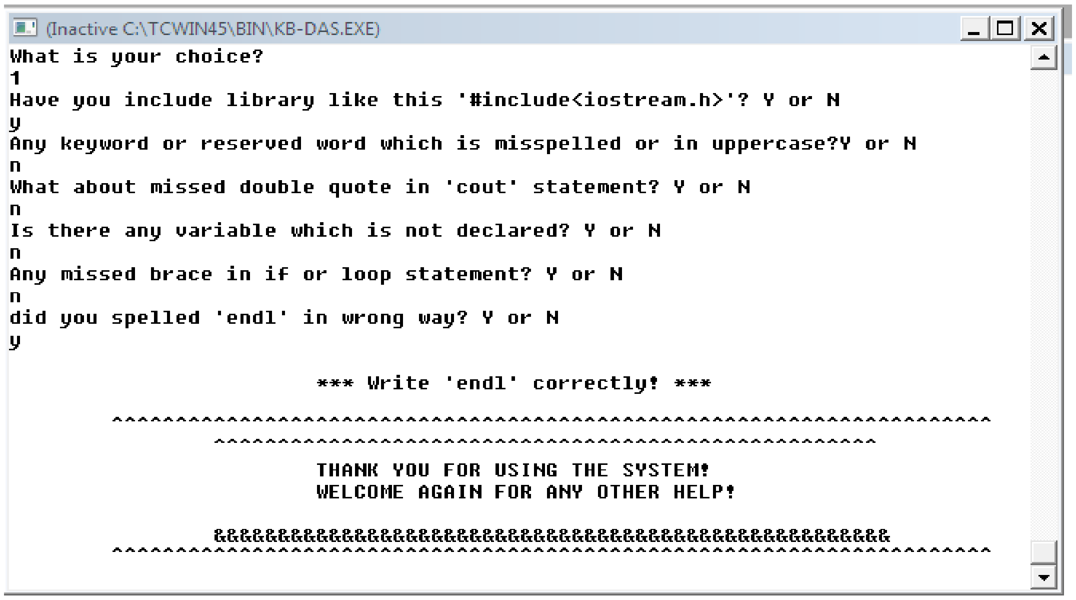

C:\TCWIN45\BIN\KB-DAS.EXE

```
        NOTE:-This knowledge based debugging assistance system is a system which
        helps novice programmers to find and correct errors while writing any
        program in turbo C++. It is developed by Tariku Fetene.

**Remind: " ERRORS are happened 'on' or 'on the top' of the highlighted line" **

 ________________________________________________________________________________

        What error message did you find? please choose from the list below.
        1. Undefined Symbol             17. Too many types in declaration
        2. Declaration Syntax error     18. Too many initializers
        3. Unable to open include file  19. Size is unknown
        4. Unknown preprocessor direct  20. Multiple declarations
        5. Unexpected }                 21. Duplicated case in function main()
        6. No file name ending          22. If statement missing
        7. Bad file name format         23. Unhandled exception (Divide error)
        8. Compound statement missing   24. General protection exception
        9. Illegal structure operation  25. Misplaced else
        10.Expression syntax error      26. Call to undefined function
        11.Statement missing            27. Linker error: undefined symbol
        12.Declaration terminated inco  28. ')' Expected
        13.Unterminated String or       29. Character constant must be one or
        14.For statement missing        30. Type mismatch in re-declaration
        15.Can not convert char to int  31. Size of 'function name' is unknown
        16.Possibly incorrect assignmen 32. Runs well but display Wrong Output
```

Figure 5.4: Welcoming window of KB-DAS

Once the welcome window of KB-DAS user interface displayed, the user can interact with the system by choosing the problem which is the user has encountered while writing C++ program. The system requests the user what error he or she has faced. If the response for this request is not given, the system does not allow the user to proceed because the system expected the user has come across with an error. However, If the user gives a choice then the system allows the user to the next request because the user has come across with a kind of error. If the system is certain that the program written has a kind of error of X, then the inference engine draws the conclusion.

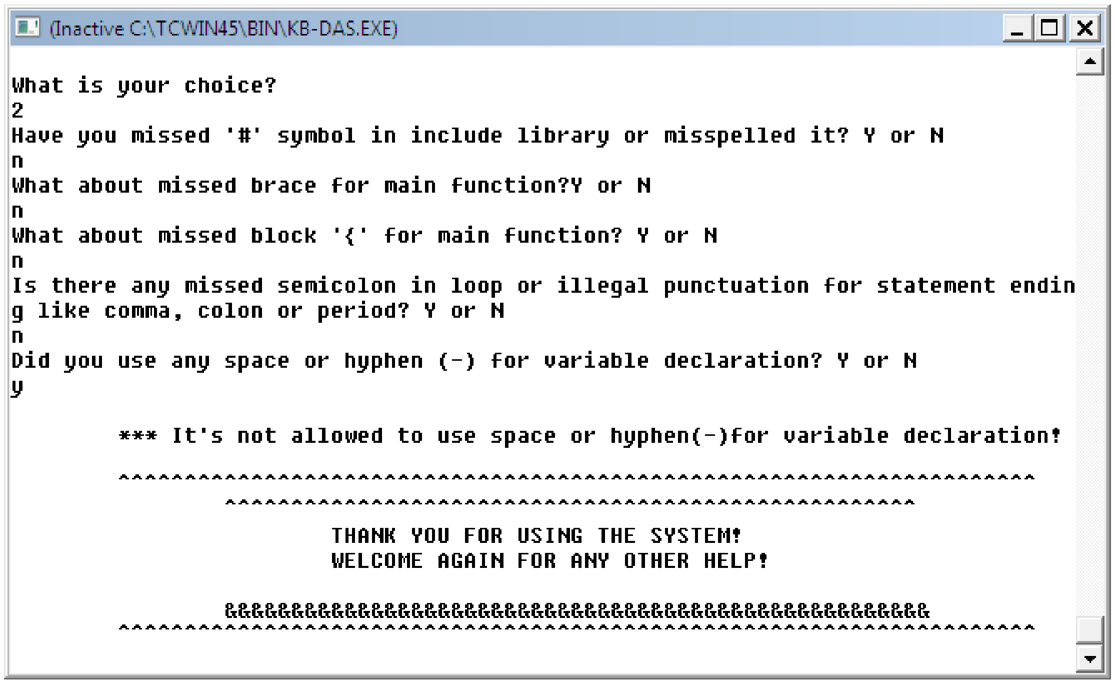
The following figure depicts the rule1, "*Undefined Symbol*", where all the conditions are satisfied. Since the undefined symbol error is displayed by the compiler because of more than six reasons so that novices may be conduct one of these reasons so the system dialog with the novice to investigate what is the most possible reasons which makes this error happened as follow:



```
What is your choice?
1
Have you include library like this '#include<iostream.h>'? Y or N
y
Any keyword or reserved word which is misspelled or in uppercase?Y or N
n
What about missed double quote in 'cout' statement? Y or N
n
Is there any variable which is not declared? Y or N
n
Any missed brace in if or loop statement? Y or N
n
did you spelled 'endl' in wrong way? Y or N
y

                        *** Write 'endl' correctly! ***

        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                        THANK YOU FOR USING THE SYSTEM!
                        WELCOME AGAIN FOR ANY OTHER HELP!

                &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Figure 5.5: dialog between user and the system on solving undefined symbol error

As already tried to express the novice user has different options to choose as per his or her desire or error faced. So that the below figure shows that when a user has encountered an error which is defined in rule 2, "*declaration syntax error*", the possible reasons would be raised to the user to have a look his or her program for correction. Here is the sample:



```
What is your choice?
2
Have you missed '#' symbol in include library or misspelled it? Y or N
n
What about missed brace for main function?Y or N
n
What about missed block '{' for main function? Y or N
n
Is there any missed semicolon in loop or illegal punctuation for statement endin
g like comma, colon or period? Y or N
n
Did you use any space or hyphen (-) for variable declaration? Y or N
y

        *** It's not allowed to use space or hyphen(-)for variable declaration!

        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                        THANK YOU FOR USING THE SYSTEM!
                        WELCOME AGAIN FOR ANY OTHER HELP!

                &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Figure 4.5: dialog on solving declaration syntax errors

CHAPTER SIX
Testing and Evaluation of the prototype

This chapter focuses on issues regarding to testing and evaluation of the prototype. This is the place where measuring the performance of the system to know the extent to which it has achieved the objective of the research or not. For the purpose of this study, KB-DAS is tested and evaluated based on the objective of the system. This is to measure the accuracy of the system during the error solving processes. In this study the performance of the system was measured against human domain expert decision in correcting errors. The user acceptance of the system was carried out during system user interaction.

The KBS user acceptance was measured by using open and close ended questions. It is used to evaluate the performance of the prototype from the users' point of view. Similarly, the questionnaires helped to assess and evaluate the acceptability and applicability of KB-DAS in the domain area. The system evaluators directly interact with the system to measure its performance from the points of its correctness in providing solutions for different problems. In addition, the validation test was done by comparing solved errors against the system conclusions on the similar issues. By comparing the result obtained from the system conclusion, the evaluators determine the performance of the system. Next to this the system has measured by using test case validation method. The evaluation questionnaires are adapted from (Pu*et al*, 2011) that used to evaluate the model called ResQue (Recommender Systems' Quality of user experience) with users' point of view. The adopted questionnaires are modified to some extent to fit them to the context of this study. These questionnaires are attached in Appendix III. System performance testing on the other hand was done by comparing the suggestions manually done by the experts with suggestions provided by KB-DAS.

10% of the previous student sample (25 in number) novice students have been selected by purposive sampling technique for user acceptance test; eight of them from computer science, seven of them from information technology and five of them from information system; Ten domain experts were selected; five of them were those instructors who already had an interview for knowledge acquisition and the rest of five were new from the departments; in evaluating the prototype system.

6.1 User acceptance evaluation

The type of questionnaires distributed for feedback collection from the evaluators was closed ended and open ended questionnaires focusing on easiness, attractiveness, time efficiency, and accuracy of the knowledge based debugging assistance system for novice C++ programmer (KB-DAS). The evaluators were allowed to rate the options using checkbox questions. The options of the check box questions are excellent, very good, good, fair, and poor for these closed ended questions. Therefore, for easiness of analyzing the relative performance of the prototype based on the user evaluation after the interaction with the system, the researcher assigned numeric value for each of the options given in words. The values are given as Excellent = 5, Very good = 4, Good = 3, Fair = 2, and Poor = 1. The Table below indicates the feedbacks obtained from the domain experts (evaluators) on systems, interaction as calculated based on the given scale. Thus, this method helps the researcher to manually examine the user acceptance based on evaluator‘s response. The average performance of user acceptance of the system is measured manually as follows:

Table 6.1: User acceptance of the system domain expert’s perspective

No	**Questions**	**1**	**2**	**3**	**4**	**5**	**Average**	**Percent**
1	Is the system more efficient in running time?	0	0	2	4	4	4.2	84
2	Does the system incorporate sufficient knowledge to solve an error which faces you?	0	0	3	3	4	4.1	82
3	Is the system accurate in analyzing facts and decision making?	0	0	2	3	5	4.3	86

4	Is the system's attractive to users?	0	0	1	3	6	4.5	90
5	Is the system's easy to use?	0	0	0	4	6	4.6	92
6	Is the system provides the right description and suggestion to be followed while finding and correcting errors by human expert	0	0	1	4	5	4.4	88
7	How do you rate the significance of the system in the domain area?	0	0	1	3	6	4.5	90
	Total						4.37	87.4

Table 6.2: User acceptance of the system novice student's perspective

No	**Questions**	**1**	**2**	**3**	**4**	**5**	**Average**	**Percent**
1	Is the system more efficient in running time?	0	0	2	11	12	4.4	88
2	Does the system incorporate sufficient knowledge to solve an error which faces you?	0	0	3	11	11	4.3	86
3	Is the system accurate in analyzing facts and decision making?	0	0	2	11	12	4.4	88
4	Is the system's attractive to users?	0	0	3	11	11	4.3	86
5	Is the system's easy to use?	0	0	1	6	18	4.7	94
6	Is the system provides the right description and suggestion to be	0	0	5	10	10	4.2	84

	followed while finding and correcting errors by human expert							
7	How do you rate the significance of the system in the domain area?	0	0	5	10	10	4.2	84
	Total						4.357	87.14

Table 6.3: total user acceptance (both teacher and novice perspective)

Perspectives	**Average**	**Percent**
Experts (teacher's)	4.37	87.4
Novice (student's)	4.357	87.14
Total	4.3635	**87.27**

As shown in the above tables, 40 % of the evaluators scored the efficiency of the system in time; how efficient the system is while interacting with the prototype system criteria of evaluation as excellent and 40% as very good , 20 % as good . The second evaluation criteria was how does the system incorporate sufficient knowledge to solve an error which faces the user and it was scored 40 % as excellent, 30 % as very good, and 30% as good. For system accuracy in analyzing facts and decision making, 50 % of the evaluators scored as excellent, 30 % as very good, and the rest 20% as good. The system attractiveness is also tested. Accordingly, 60% experts are satisfied with the interface which is scored excellent, 30% of them selected very good and 10% are in good mood with interface attractiveness. It is scored 60% excellent and 0% very good regarding to it easiness to use sine it presents with the environment which fundamentals of programming course is delivered.

Moreover, 50 % of the evaluators gave the prototype system an excellent score with regard to the system provides the right description and suggestion to be followed while finding and correcting errors by human expert40 % as very good, and 10% as good. The significance of the knowledge based system to assist debugging for novices was rated by 60 % of respondents as excellent while 30 % rated the prototype system as very good and 10 % as good. Finally, the average performance of the prototype system according to the evaluation results filled by the domain experts is 4.37 out of 5 4.357 by novice students and total is 4.3635 which is 87.27 %.

In addition to the closed ended questions, the evaluators were provided with open ended questions to forward their suggestions and opinions. These questions focus on how the KB-DAS differs from the teacher experts in processing debugging codes written by turbo C++.

The *first* open-ended question the respondents were asked was to know how is KB-DAS different from a debugging style conducted by human expert. All respondents agreed on that the system differs from the style which human expert can debug a program, it asks lots of questions which are the most possible cause for the single error message so that novices can learn lots of error causes for a single error message and again for the their next program they may remember the cause for an error which is come across when they debug another error. But human expert do not provide the possible reasons for that single error but instead they go to find the error and correct it this time novice students think that error is the only reason for the error they encountered but in reality it is wrong since the compiler notifies a single error message for may be lots of reasons. For instance for a single error message "*statement missing*" there may be lots of reasons like there is any punctuation instead of semicolon, there is any missed semicolon or illegal punctuation at end of statement such as period, comma or colon, there is no <<for cout statement, there is any missed brace in if statement, there is missed double quote in cout statement, there is double quote in cout statement repeatedly without backward slash (\) and so on. All these are the possible reasons for the error statement missing.

Another difference in between the system and human expert is that human intervention cannot be 24/7 or in the time they practice their own code personally. So this kind of debugging assistance system helps novices to get involved with it for their problems concerning to debugging.

The *second* open ended question raised was “*what makes KB-DAS different from the websites available to consult debugging*” respondents answered this question by raising the problems of those websites. One of the big problems for most websites is there is no much focus on debugging but rather on concepts tutorial. Actually there are some Q&A, forums, and blogs which are socialized but we may not get specific answer for specific question but sometimes if we are luck and similar case is already asked then we may get solution. Another big problem for novices is they do not know those websites name since they are may not be oriented by their teachers. So, searching for good Q&A, forums and blogs takes too much time for their specific question. Another challenge for those websites is, they only answer for frequently asked questions and in case the error which novice faced is new they have no solution. However, such a kind of system (KB-DAS) will be encouraged and different from websites.

In the general, all respondents agreed that the system can really assist novices in debugging process and encourage them well to be a mover in their program writing skill, but it might need further development because the errors which is committed by novices are very complicated and too much.

The *third* question was “*Does the system have any significance in the domain area*” and accordingly respondents answered it as the system has significance in the domain area. All the system evaluator‘s responded that the system add value in the domain area. In addition the system also can reduce the burden of human expert by saving their time and energy spent while debugging especially for large class size in where many students has come across with errors in their program ad since it is very difficult for teachers to be always there for students 24/7, the system can substitute teacher when the need come to novices.

The *fourth* question was about the significant strength of the system and accordingly it is mentioned as it saves the user’s time and energy. It does not expect every input from the user because it does its own internal processing by asking the possible chance of the error to take the load off the user. The system makes accurate and reliable decisions and reduces the chance of errors in writing C++ program. The perceived limitations of the system are that the types of error limitation and user interface of the system needs improvement to make the system more attractive to users. Another limitation raise was since the system interacts with the user using only ‘yes’ or ‘no ‘replies. Therefore, it lacks some flexibility.

6.2 System evaluation using Test Cases

In the user acceptance evaluation, it is discussed about the evaluation of system performance using both closed and open ended questions. System evaluators directly interacted with system using these questions in order to forward their feedback and suggestion on the performance of the system.

In this section the performance of the system was tested and validated using test cases. The test cases were used to measure the accuracy of the system. For the purpose of validation process a total of twenty five cases were selected. To achieve the goal of the system evaluators were purposively selected according to their willingness and specialty.

The KB-DAS testing procedure was carried out by system evaluator to evaluate the solutions suggested by the system were correct or incorrect. System evaluators compared the decisions made by the system against human expert. Then system evaluators validated the number of correct decisions made by the system. The result of the comparison shows that the rule based system has made close decision in the debugging process of problems as human experts did. As indicated in table 6.4 below, the result provided by system evaluators showed that the knowledge based system is about 75% accurate in debugging computer program errors written in turbo C++.

Table 6.4: Testing the Accuracy of KB-DAS by using test cases

Errors	**Total number of errors selected**	**Correct suggestion**	**Incorrect or no suggestion**	**The accuracy of the prototype in %**
Undefined symbol	4	3	1	75
Declaration syntax error	4	4	0	100
Compound statement missing	3	3	0	100
Expression syntax error	4	3	1	75
Statement missing	4	3	1	75
For statement missing	3	3	0	100
Runs well but display wrong output	5	2	3	40
Total	25	20	5	80.0

From table 6.4 above twenty five cases, which incorporated with errors novice experiencing, were used to validate the accuracy of the system. For any errors stored in the knowledge base, the knowledge base system can suggest solutions. Purposively selected errors are used to challenge the system performance. As a result, for "*undefined symbol errors*" in the above table 6.4 from the given four cases three of them are correctly suggested by the system (75%). This simply shows that there are other factors which make undefined symbol error happened rather than what we have acquired from experts which is forwarded for further research. Similarly, from the given four cases all of them are classified correctly in the "*declaration syntax errors*" (100%). For the "*compound statement missing*" errors while writing computer program in turbo C++ the system correctly provided suggestion (solutions) for all the errors (100%). Out of four presented cases for an error "*expression syntax error*" three of them are correctly suggested by the system which means the system is 75% accurate. The same is true for "*statement missing*" errors which 75% accurate. *"For statement missing*" registered as 100% accurate but the "*runs well but wrong output*" logical error registered below half which is 40%. This simply implied that there needs additional effort on logical and run time errors.

Finally, the result indicated that all the cases are directly similar with knowledge incorporated in the knowledge base and average performance of the KB-DAS is 80%. It reveals that the compile time errors have addressed well but further research should be done on logical and run time errors. Finally, sample of errors and cases are attached at the **appendix VI**.

6.3 Comparison with related works

Table 6.5: Dealing with the previous related works with the current system

Related works	**Done for**	**Author**	**Success**
Novice Assistancein Java Introduction	Java novices	Salcedo, NajinarRaysal Marie G	The system has gained User acceptance and system performance is not specified
DebugIt	Pascal language	Lee and Wu	94.75% accurate and user acceptance does not registered
Recent Progress in the Development of	C language	Smith and Webb	76.4% accuracy and user acceptance is unspecified

a Debugging Assistant for Computer Programsor "*Bradman"*			The limitation of Bradmanstatements are treated as individual entities and no attempt is made to understand their purpose in relation to other statements.
Expresso	Java language	Bryn Mawr College	86.75% accuracy registered. No specified user acceptance. The tool specifically does not eliminate the need for understandable compiler error messages; rather, the tool enhances the functions of a compiler.
Adil (Automated Debugger in Learning system)	C language	John F, Alex Myth, Watson G.	Given a syntax error-free program and its specification, this debugger is able to locate, pinpoint and explain logical errors of programs
HelpMeOut (social recommender system)	It collects examples of code changes that fix errors in a central database	Gate H, Kenedy J, Mark F	The system is able to suggest useful fixes for 47% of the errors.
Java Intelligent Tutoring System	It is a web-based application where you will upload your	Daniel T, Wiston B and Criss R.	Presented as it is accurate in user acceptance and system

	java program and run your program and returns the output		performance but not numbered in percentage
KB-DAS (knowledge base debugging assistance system)	C++ language	Tariku Fetene	87.27% user acceptance and 80 % accuracy performance

So, the prototype knowledge based debugging assistance system for C++ program can be concluded as promising and applicable in the domain area. The feedback and suggestion of domain expert reveals that the knowledge based system satisfactorily gained user acceptance. The system acceptance evaluations used open and close ended questions to directly interact with system and registered good accuracy.

CHAPTER SEVEN
Conclusion and Recommendations

7.1 Conclusion

In teaching and learning programming ideas and skills in finding errors and correcting them to have an error free program has been recognized as a great challenge for novice programmers. The primary objective of this study was to develop knowledge base system that supports novice programmers in debugging computer program written in C++.

The study focused on novice programmer, according to Dreyfus (1996) categorization of programmers novices are the very beginner programmers who has no any previous knowledge in any kind of computer programming others are advanced beginners, competent, proficient and expert at the last. This is because it was found as logical to start them early case and again it is supported by scholars those who are trained early in debugging would become better debuggers more quickly (Hwang *et al*, 2012).Debugging training is even more needed by novice programmers (Oman *et al*, 2011).

To achieve the objective of the study, the following research questions were raised to be answered:

- ❖ What are the challenges for both teachers and students, in teaching learning computer programming?
- ❖ What are the common errors novices are experiencing how can these problem be solved?
- ❖ What knowledge is there in programming style and coding conventions to write error free program?
- ❖ To what extent the application of knowledge based system support novice programmers in debugging computer program?

Since, over the years, the discipline of knowledge engineering has evolved into the development of theory, methods and tools for developing knowledge-intensive applications (Marcus and McDermott, 2011). So, in this research it is employed a Knowledge engineering method for knowledge acquisition, model building, representation and prototype development and testing, whereas other suitable methods are also used for knowledge elicitation through discussion with experts which are professional and experienced teachers and survey design to assess the level of students in understanding compiler error messages, the way how students debug their program and teachers reaction in assisting them. Survey design is more effective in assessing the current practices in its natural setting (Best & Kahn, 2003).

Accordingly, the following were the major findings of the study:

- ❖ Course Complexity (multidisciplinary), Students' motivation, way of study, methodology and tools used traditional teaching methods, normally based on lectures and specific programming language syntaxes, often fail in what concerns the students' motivation in getting involved in meaningful programming activities, Student's previous experiences in any kind of programming, source code developmental tools (unclear compiler error message), natural language problem are found as the main reasons for students' failure in computer programming courses.
- ❖ Different common errors novices are experiencing have explored such as missing library, variable not declared, variable and data type mismatch. For more common errors *(page 56-61 of this thesis*).
- ❖ Knowledge which exists in programming style and coding conventions to help in writing error free program in turbo C++ are also explored so that they could be used as an input for the knowledge base system.
- ❖ Another major finding is that the more specific errors messages from the compiler could help students to clarify concepts, misconceptions, or improve their debugging experience so that their level of understanding for compiler errors message would be high and they can easily find and correct the errors.
- ❖ Since compilers are not always helpful, with the help of KB-DAS, knowledge base debugging assistance system that addresses the needs of novice programmers. It implemented the about 89 errors identified. The more specific

messages from the KB-DAS help students to become better programmers in terms of debugging and writing error-free programs by improving their program comprehension skills and giving them debugging experience

- ❖ Applicability of KBS for debugging computer programs has been proved. And the prototype knowledge based system is promising and applicable in the domain area. The feedback and suggestion of domain expert reveals that the knowledge based system satisfactorily gained user acceptance. But, to fully provide debugging some types of error especially logical and run time errors need to be incorporated into the current system.

7.2 Recommendations

The prototype knowledge based debugging assistant system is promising and applicable in the domain area of debugging computer program written in turbo C++ and the feedback and suggestion of domain expert reveals that the knowledge based system gained user acceptance and tested its performance so that it is highly recommended for school of informatics use the system for improving students skill in debugging.

In the system developed, 89common errors (35 compiler error messages), in which there are on average three types of errors (syntax, logical and run time) novices do, were handled which are acquired from teacher experts. So that the prototype is developed by having these 35 production rules however it would be better to add more errors in order to make the system inclusive of all the errors committed by the novice programmers in turbo C++.

Most of the errors tried to represent into the knowledge base system are compile time errors so that it needs further investigation for logical error which is another headache for novice and other higher experts. It is recommended the scope of the knowledge based system should be extended to incorporate other logical and run time errors.

In this study an attempt is made to apply rule based systems. But, there are different solved cases available in computer program error debugging especially logical errors.

Rule based systems solve problems from scratch, while case based systems use pre-stored situations to deal with similar new instances. Therefore, the integration of rule based reasoning with case based reasoning would solve the limitation when representing knowledge in the form of if then rules unable to draw a conclusion for logical errors.

It is also recommended to have an alternative idea to use specialized tools to log students' actions in order to easily explore their behavior while programming and compiling, which can provide good insight into which students are facing most problems when, and guide the instructors consequently. Or in other word it means a tool that not only collects actual compilation errors but also prepares reports both for instructors and students with suggestions and recommendations.

Another recommendation is goes to compiler designers to think over and add a feature of automatic keyword options in turbo so that it would be possible to decrease the probability of novices to experience some simple errors which happens because of keyword usage and misspelling.

References

Aamodt, A., Plaza, E. (2013). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. IOS Press, Vol. 7: 1, pp. 39-59

Aboneh, T. (2013). Knowledge based system for pre-medical triage treatment at Adama University Asella Hospital.‖ Masters Thesis, Addis Ababa University, Ethiopia, 2013.

Abdulah,M., Paige, R., Thpmpson, I., Benes, C. (2014). Conceptual modeling of knowledge Based Systems by Using UML‖ , pp. 24-33, Sawed Arabia.

Ahmadzadeh, M., Elliman, D and Higgins, C. (2005).An analysis of patterns of debugging among novice Computer Science students, SIGCSE, pp. 84-88.

Akinola, K. E.,Olanrewaju, G. O., Oyenuga, A. Y.(2015).Improvement Strategies for Computer Science Students' Academic Performance in Programming Skill.American Journal of Computer Science and Information Engineering. Vol. 2, No. 5, 2015, pp. 45-50..

Aljunid, S., Nordin, M., Shukur, Z. and Zin, A. (2000) A Knowledge-based Automated Debugger in Learning System

Bain, K. (2004). What the best college teachers do. Cambridge, MA: Harvard University.

Bradnt, J., Hartmann, B., Klemmer, S, and MacDougall, D. (2010). What Would Other Programmers Do? Suggesting Solutions to Error Messages.

Deek, F., Kimmel, H. & McHugh, J. (2014). Pedagogical changes in the delivery of the first-course in computer science: Problem solving, then programming. Journal of Engineering Education, 87, pp. 313-320.

Dewey, J. (1997) Experience and Education.Free Press, New York, 1997 (reprint edition).

Eckerdal, A. (2009). Novice programming students' learning of concepts and practise(Doctoral dissertation), Retrieved from http://uu.divaportal.org/smash/record.jsf?pid=diva2:173221

Endris, U. (2011). An Introduction to Prolog Programming.Amsterdam University, van Amsterdam.

Freeman-Hargis, J. (2014). Methods of Rule-Based Systems. Retrieved from http://ai-depot.com/Contest/Rule- Based Systems and Identification Trees

Hristova, M., Misra, A., Rutter, M and R. Mercuri.(2003). Identifying and correcting programming errors for introductory computer science students," ACM SIGCSE, pp. 153-156.

Hwang, W. Y., Shadiev, S., Wang, C. Y., & Huang, Z. H. (2012).A pilot study of cooperative programming learning behavior and its relationship with students' learning performance.Computers & Education, 58, 1267–1281.

Katz, I. and Anderson, J. (1987).Debugging: An analysis of bug location strategies.Human-Computer Interaction, 3, 4, 351-399.

Kesarwani, P., &Misra, A. (2013).selecting integrated approche for knowledge representation by comparative study of knowledge representation schemes. International Journal of Scientific and Research Publications, 1-5.

Kingston, J. (2008). Knowledge based system development tools. Artificial Intelligence, University of Edinburgh, Scotland.

Kolling, M. & Rosenberg, J. (2013). Blue - A Language for Teaching Object-Oriented Programming, Proc. of the 27th SIGCSE Technical Symposium on Computer Science Education, pp. 190-194.

Kordaki, M. (2010). A drawing and multi-representational computer environment for beginner learning of programming using C: Design and pilot formative evaluation. Computers & Education, 54, 69–87.

Kurland, D.M., Pea, R.D., Clement, C., Mawby, R. (2013). A study of the development of programming ability and thinking skills in high school students. In: Soloway, E., Spohrer, J.C. (Eds.), Studying the Novice Programmer. London, Lawrence Erlbaum Associates, 83–112.

Lee, G. C., & Wu, J. C. (1999). Debug It.Computers& Education, 32(2), 165-179.

Lewis, M and Gregg,C. (2016). How do you teach debugging?: resources and strategies for better student debugging, ACM SIGCSE, pp. 706-706.

Maher, M. L. (1984). An Expert System For The Preliminary Structural Design Of High Rise Buildings, Forthcoming Ph. D. thesis, Department of Civil Engineering, Carnegie-Mellon University.

Marcus, S and McDermott, J. (2011). A knowledge acquisition language for propose and- revise systems. Artificial Intelligence, 39(1):1–38.

McCartney, A.,Eckerdal , J. E.,Mostrom , K., Sanders , C. (2007). Successful students' strategies for getting unstuck, The 12th annual SIGCSE conference on Innovation and technology in computer science education, 2007, Dundee, Scotland.

McCracken, J.,Mayer, R., Dyck, J. &Vilberg, W. (2014). Learning to program and learning to think: what's the connection? In Soloway&Spohrer: Studying the Novice Programmer, pp. 113-124

Mulder, F. (2002). Computer Science: from a BÈTA to a DELTA subject. Informatica, Tinfon, 11, 48.

Nalepa, G. (2015). Methodologies and Technologies for Rule-Based Systems Design and Implementation.Towards Hybrid Knowledge Engineering.AGH University of Science and Technology, Poland.

Olapiriyakul, K. &Scher, J. M. (2012). A guide to establishing hybrid learning courses: employing information technology to create a new learning experience, and a case study. The Internet and Higher Education, 9, 287–301.

Oman, P. W., Cook, R., &Nanja, M. (2011). Effects of programming experience in debugging semantic errors.Journal of Systems and Software, 9(3), 197-207.

Perkins, D., Hanconck, C., Hobbs, R., Martin, F. & Simmons, R. (2014). Conditions of learning in novice programmers. In Soloway&Spohrer: Studying the Novice Programmer, pp. 261-279.

prentzas, j., &Hatzilygeroudis, l. (2007). Categorizing approaches combining rulebased and casebased reasoning. Exprt systems, 24(2), 97-122.

Pu, P., Chen, L., & Hu, R. (2011).A User-Centric Evaluation Framework for Recommender Systems.5th ACM Conference on Recommender Systems , (pp. 157 – 164). Chicago.

Rajeswari, P. V. (2012). Hybrid Systems for Knowledge Representation in Artificial Intelligence.International Journal of Advanced Research in Artificial Intelligence.

Robert-Jan Mora and Bas Kloet. (2010). The application of statistical sampling in digital forensics (Vol. Version:1.0). Hoffmann Investigations, Almere The Netherlands.

Sajja, P., &Akerkar, R. (2010).Knowledge-Based Systems for Development.Advanced Knowledge Based Systems:Model, Applications & Research (Vol. I, pp. 1 – 11).

Salcedo, G. (2016). Novice Assistance in Java Introduction.University of the philippinesmanila , college of arts and sciences , department of physical sciences and mathematics.

Schulte, C. &Bennedsen, J. (2006).What do teachers teach in introductory programming? In Proceedings of the Second International Workshop on Computing Education Research, Canterbury, UK, September 9–10, (pp. 17–28).ICER ‘06. New York: ACM.

Sharma, T., &Kelkar, D. (2012). A Tour Towards Knowledge Representation tecniques. International Journal of Computer Technology and Electronics Engineering, 131-135.

Sykes, E., & Franek, F. (2004).A Prototype for an Intelligent Tutoring System for Students Learning to Program in Java.Advanced Technology for Learning, 1(1), 1-6.

Smith, P and Webb G. (1992).Recent Progress in the Development of a Debugging Assistant for Computer Programs

Thomas, B. (2014). Evaluation of Knowledge based systems. What can be evaluated and what cannot? Journal of Evaluation in Knowledge base system, Vol. 4, pp. 373–385, US.

Verdú, E., Regueras. L. M., Verdú, M. J., Leal, P. J., Castro, J. P., &Queirós, R. (2012). A distributed system for learning programming online.Computers & Education, 58, 1–10.

Wang, Y., Li, H., Feng, Y., Jiang, Y., & Liu, Y. (2012). Assessment of programming language learning based on peer code review model: Implementation and experience report. Computers & Education, 59, 412–422.

Winslow, L.E. (2013). Programming pedagogy - A psychological overview. SIGCSE Bulleting, 28(3), pp. 17-22.

APPENDIXES

Appendix I

Pre survey of student's grade reports (below C) for consecutive three year in programming courses

		Below Grade C										
Courses	Year	C S	IT	IS	Physics	Statistics	Mathematics	Civil Eng.	Mech Eng.	Elec Eng.	Total (%)	
Fundamental of programming I	2006E.C	67 / 97	61/93	27/43	31/56	21/35	19/43	76/104	81/121	69/98	62.20%	66.25%
	2007E.C	61/88	67/91	21/38	38/62	29/42	23/36	59/108	62/114	47/78	63.53%	
	2008E.C	81/101	69/90	30/45	38/57	26/40	22/33	89/117	67/101	51/73	73.02%	
Fundamental of programming II	2006E.C	59 / 81	57/84	21/39							67.15%	67.62%
	2007E.C	38/82	61/82	22/36							60.5%	
	2008E.C	77/95	64/88	29/43							75.22%	
	2006E.C	54/79	49/80	23/37						43/85	60.14%	

Object oriented programming	2007E.C	56/80	58/78	19/37						31/82	59.21%	58.09%
	2008E.C	61/90	53/86	18/43						30/76	54.91%	
Advanced programming	2006E.C	54/72	48/78	23/39						39/75	62.12%	59.92%
	2007E.C	56/70	48/66	21/37						31/80	61.66%	
	2008E.C	51/67	43/79	19/41						33/74	56%	
Visual Basic	2006E.C		41/77	18/39						41/75	52.35%	51.46%
	2007E.C		39/71	21/37						31/80	48.4%	
	2008E.C		43/79	23/41						38/74	53.62%	

Appendix II

Jimma University

College of Natural Science

Department of Information Science

Questionnaire for the fulfillment of masters program in information science

First of all I would like to thank you for your cooperativeness to help me in assessing the overall status in teaching learning computer programming and also I appreciate your genuine response. Then, kindly I will ask you to fulfill the following requirements. "Thanks in advance"

For students in Mizan Teppi University

1. Sex________
2. Department ______________________________
3. Year I ☐ II ☐ III ☐ IV ☐ V ☐
4. How is your interest in programming courses? Do you like programming courses? Yes ☐ No ☐
5. If No, why?

   ____

   ____.
6. How about your grades in programming courses on average relative to other courses? (**Very high** A+ & A, **high** A-,B+ &B, **fair** B-,C+ &C, **low** C-,D and Fx **very low** F) Very High ☐ High ☐ Fair ☐ Low ☐ Very low ☐
7. If your grade is not very high or high, what do you think is the reason?

☐	Course complexity	☐	Developmental tool
☐	Teaching methodologies and strategies	☐	Natural language problem
		☐	Previous experience
☐	Student's motivation	☐	Way of study

8. How do you understand the message, whether it is warning or errors, notified by the compiler while writing a computer program?

Very easily ☐ Easily ☐ Neutral ☐ Not easily ☐ even can't understand ☐

9. Teacher's teaching frame of mind is always the same

Strongly agree ☐ agree ☐ no idea ☐ disagree ☐ strongly disagree ☐

10. What kind of problem did you observe by teachers while teaching computer programming course?
    - ☐ Bored
    - ☐ Tiresome
    - ☐ Not subject knowledgeable
    - ☐ Carelessness
    - ☐ Incomprehensible
11. If there is one or more teacher who is your favorite in teaching computer program, what is the reason?
    - ☐ Teach the course easily
    - ☐ They are friendly
    - ☐ They are encouraging
    - ☐ They are understandable
12. How do you see yourself? Are you stopper or mover (In problematic situation stoppers simply stop and abandon all hope of solving the problem on their own, while movers keep trying, modifying their code and use feedback about errors effectively.)?

    Stopper ☐ Mover ☐

13. Where is the conceptual problem in writing computer programming?
    - ☐ Source code writing
    - ☐ Programming conceptual understanding
    - ☐ Completing incomplete code
    - ☐ Outputting for fragment code
    - ☐ Logical design
    - ☐ Compiler error correction
    - ☐ Running time error correction
14. Please rate the conceptual difficulty or challenge for you in writing computer programming?

a.	**Variable**&**data type**	☐ High	☐ Medium	☐ Low
b.	**Conditional statement**	☐ High	☐ Medium	☐ Low
c.	**Loop statement**	☐ High	☐ Medium	☐ Low
d.	**Array**	☐ High	☐ Medium	☐ Low
e.	**Pointer**	☐ High	☐ Medium	☐ Low
f.	**Modular programming**	☐ High	☐ Medium	☐ Low

g.	**Structure statement**	☐ High	☐ Medium	☐ Low
h.	**File system**	☐ High	☐ Medium	☐ Low
i.	**Syntax**	☐ High	☐ Medium	☐ Low
j.	**OOP concepts**	☐ High	☐ Medium	☐ Low
k.	**Exception handling**	☐ High	☐ Medium	☐ Low
l.	**Other** ☐			

15. Where do you frequently spend your time to study your programming courses?
    - ☐ Trying codes in laboratory
    - ☐ By reading only hand out independently
    - ☐ Peer Group study
16. How much time on average do you spend in studying programming courses (per day)?
    - ☐ Less than 1 hour
    - ☐ Up to 2 hours
    - ☐ Up to 4 hours
    - ☐ Above 4 hours
17. Where do you try so as to find answers for every question you have in writing programming?
    - ☐ From teachers
    - ☐ From students
    - ☐ From development environment or tool
    - ☐ From internet
    - ☐ From books
    - ☐ No place
18. What is the reason for preferring to solve your challenges by the method you already checked for the above questions (question number 17)
    - ☐ Easy to access
    - ☐ Easy to use
    - ☐ Easy to understand

Appendix III

Jimma University

College of Natural Science

Department of Information Science

Interview checklist for teachers for the fulfillment of masters program in information science

First of all I would like to thank you for your cooperativeness to help me in assessing the overall status in teaching learning computer programmingand also I appreciate your genuine response. Then, kindly I will ask you to fulfill the following requirements.

"Thanks in advance"

For teachers in MizanTeppi University

1. Sex_________
2. Department _____________________________
3. Educational status
   PHD ☐ MSc ☐ BSc G-3 ☐ BSc G-2 ☐ BSc G-1☐
4. For how long have you stayed in teaching?
   1-2 year☐2-4 years☐4-6 years☐6-10 years☐above 10 years☐
5. Have you ever provided programming courses? Yes ☐ No ☐
6. What do you think the reason why students score low in computer programming courses?
7. How computer program debugging can be defined?
8. How bugs happen in writing computer programming?
9. How these bugs could have effect on student's practical exercise?
10. Are these bugs can be debugged?
11. What are the general types of errors that are common across different programming languages and those that are particular to C++, the language the novice programmers used?
12. When students debug their program, what do they do when they get stuck?
13. What kind of consult do you recommend to improve student's debugging skills which can also assist teachers in teaching learning computer program?

Appendix IV

Production rules

Rule 1: undefined symbol, if

There is no library;

There is misspelled keyword or keyword in uppercase;

There is any missed double quote in cout statement;

There is any used variable which is not declared first;

There is any missed brace in if or loop statements;

‘endl’ is not correctly spelled,

Rule 2: declaration syntax error, if

There is missed # in libraries and misspelled;

There is missed brace in main function;

There is any missed block for main function;

There is any missed semicolon in loop or illegal punctuation for statement ending like comma, colon or period;

There is any wrong variable declaration like space or hyphen.

Rule 3: unable to open include file, if

There is missed .h in importing libraries.

Rule 4: unknown preprocessor directive, if

There is library is not correctly written like missing <>;

There written include in uppercase.

Rule 5: unexpected }, if

There is a missed open block.

Rule 6: no file name ending, if

There is a missed > in iostream.h.

Rule 7: bad file name format, if

There is a missed < in iostream.h.

Rule 8: compound statement missing, if

There is any missed closed block;

There is any missed semicolon or illegal punctuation at end of statement such as period, comma or colon;

There is missed double quote in cout statement;

{ or [symbol is used instead of brace for if and loop statements.

Rule 9: illegal structure operation, if

>> is used for cout statement and << for cin statements.

Rule 10: expression syntax error, if

>>>or<<< or<<> or other wrong way is used for cin and cout statements;

One **<** is missed in cout statement when concatenating two or more statements;

Comma, colon or period is used instead of semicolon;

+ instead of ++,<== instead of <=, >== instead of >= is used in if or loop statement;

There is any wrong expression in if or loop statement;

There is wrong array initialization;

There is unnecessary symbol like **|** or / for comment.

Rule 11: statement missing, if

There is any punctuation instead of semicolon;

There is any missed semicolon or illegal punctuation at end of statement such as period, comma or colon;

There is no <<for cout statement;

There is any missed brace in if statement;

There is missed double quote in cout statement.

There is double quote in cout statement repeatedly without backward slash (\)

Rule 12: declaration terminated incorrectly, if

There is any variable begins with number or symbol;

There is any missed semicolon or illegal punctuation at end of statement such as comma, period or colon;

Rule 13: unterminated string or character, if

We initialize character for int data type;

Missed " for closing cout statement like cout<<"Hello world;

Rule 14: for statement missing, if

Missed semicolon in loop statement;

Only + used for incremental statement ++;

Wrong expression in loop statement;

Missed brace or other symbol used like [ ] or { } for loop statement;

Missed initialization, condition and increment for looping statement.

Rule 15: cannot convert char to int, if

“ is used for int variable. Egint age = “18”

Rule 16: possibly incorrect assignment, if

= is used instead of == for equal statement.

Rule 17: too many types in declaration, if

Data type is used as a variable.

Rule 18: too many initializers, if

Too many values are initialized for array, which is out of the index;

There is no closed block “}” for array initialization;

Wrong use of block for array initialization such as [ ] or ( ).

.Rule 19: size is unknown, if

Do not initialize the size of an array.

Rule 20: multiple declarations, if

There are similar variable name with the same data type or even different data type.

Rule 21: duplicated case in function main(), if

There is the same case in switch statement.

Rule 22: if statement missing, if

There is missed brace in IF, either open brace ‘(’ or closed ‘)’ .

Rule 23: unexpected output, if

There is incorrect condition in loop of if statements.

Rule 24: unhandled exception (Divide error exception), if

A number is divided by 0.

Rule 25: general protection exception, if

One **<** is missed in cout statement when concatenating two or more that two statements egcout<<”hello”**<**”world”.

Rule 26: misplaced else, if

There is no closing block ‘}’ for IF but used else.

Rule 27: additional ‘0’ output for array program, if

The size of the array or index is more than the values initialized at first.

Rule 28: unwanted number output, if

The compiler gets an output which is array index out of bound especially when loop is used and data type of the array is int, float or double.

Rule 29: unwanted character output, if

The compiler gets an output which is array index out of bound especially when loop is used and data type of the array is character.

Rule 30: type mismatch in re-declaration, if

The data type of a function definition is different from its declaration.

Rule 31: size of ‘function name’ is unknown or 0, if

hyphen is used for a function name while declaring.

Rule 32: call to undefined function, if

There is undefined function used in calling;

Function defined and called is different in naming.

Rule 33: linker error: undefined symbol, if

Trying to define a function which is not declared first;

main() function is in uppercase like this Main()

Different Function name in declaration and definition is used.

Rule 34: ‘)’ Expected, if

There is extra open brace for function declaration or IF or even loop.

Rule 35: character constant must be one or two character long, if

‘ is used instead of “ in cout statement

Rule 36: misplaced break, if

The compiler encountered a break statement outside a switch or looping.

Rule 37: misplaced continue, if

The compiler encountered a continue statement outside a looping.

Rule 38: do statement must have while, if

Your source file contained a do statement that was missing the closing while keyword

Appendix V

Evaluation questionnaires

Questionnaire to test and validate the performance of the knowledge base debugging assistance system (KB-DAS) for novice programmers:

1. Is the system more efficient in running time?

☐ Poor ☐ Fair ☐ Good ☐. Very good ☐ Excellent

2. Does the system incorporate sufficient knowledge to solve an error which faces you?

☐ Poor ☐ Fair ☐ Good ☐. Very good ☐ Excellent

3. Is the system accurate in analyzing facts and decision making?

☐ Poor ☐ Fair ☐ Good ☐. Very good ☐ Excellent

4. Is the system's attractive to users?

☐ Poor ☐ Fair ☐ Good ☐. Very good ☐ Excellent

5. Is the system's easy to use?

☐ Poor ☐ Fair ☐ Good ☐. Very good ☐ Excellent

6. Do you believe that KBDAS can effectively handle debugging processing?

☐ Yes ☐ No

7. Is the system provides the right description and suggestion to be followed while finding and correcting errors by human expert.

☐ Poor ☐ Fair ☐ Good ☐. Very good ☐ Excellent

8. How do you rate the significance of the system in the domain area?

☐ Poor ☐ Fair ☐ Good ☐. Very good ☐ Excellent

9. Does the system update its knowledge base?

☐ Poor ☐ Fair ☐ Good ☐. Very good ☐ Excellent

10. How is KB-DAS different from a debugging style conducted by human expert?

____
____
____
____.

11. What makes KB-DAS different from the websites available to consult debugging?

____
____
____.

12. Does the system have any significance in the domain area?

____
____
____.

13. What is the strength of KB-DAS?

____
____
____.

14. What are the limitations of KB-DAS?

____
____
____.

Appendix VI

Sample of Test Case Used to Validate the Accuracy of KB-DAS System

Case1: Undefined symbol error (no library)

```
void Main(){
cout<<"hello World";
}
```

Undefined symbol error (capitalized keyword 'Cout')

```
#include<iostream.h>
void main(){
Cout<<"hello World";
}
```

Undefined symbol error (missed double quote (") for cout statements)

```
#include<iostream.h>
void main(){
cout<<hello World";
}
```

Undefined symbol error (undeclared variable)

```
#include<iostream.h>
void main(){
cout<<sum;
}
```

Case2: declaration syntax error (missed # for including library)

```
include<iostream.h>
void Main(){
cout<<"hello World";
}
```

declaration syntax error (missed brace ’(’ for main function)

```
include<iostream.h>

void Main ){

cout<<"hello World";

}
```

declaration syntax error (missed block ‘{’ for main function)

```
include<iostream.h>

void Main ()

cout<<"hello World";

}
```

declaration syntax error (use hyphen for variable declaration or miss semicolon at the end of the statement)

```
include<iostream.h>

void Main( ){

int person-age

cout<<"hello World";

}
```

Case3: compound statement missing (missed closed block ‘}’)

```
#include<iostream.h>

void main(){

cout<<"Hello World";
```

compound statement missing (using ‘{’ instead of ‘(’ for IF statement)

```
#include<iostream.h>
void main(){
int a=5;
if{a>4)
cout<<"Hello World";
  }
```

Case 4: Expression Syntax Error(using extra '<' for cout statement)

```
#include<iostream.h>

void main(){

cout<<<"Hello World";

  }
```

Expression Syntax Error(using wrong operation for example '>==')

```
#include<iostream.h>

void main(){

int a=5;

if(a>==4)

cout<<"Hello World";

  }
```

Expression Syntax Error (using '[' instead of '{' for array initialization)

```
#include<iostream.h>

void main(){

int a[]=[1,2];

cout<<"Hello World";

  }
```

Expression Syntax Error (using '||' or '/' instead of '//' for comment)

```
#include<iostream.h>

void main(){

int a[]=[1,2];

||cout<<"Hello World";

  }
```

Long fragment of code with 12 errors

```
void Main(){
int b;
int a[]=[1,2,3,4,5];
cin<<b
cout<<"hello World";
cout<<'it is to check';
Cout<<"what can be the problem?";
cout<<hello World";
||cout<<"Hello World";
cout<<sum;
if{b>4)
cout<<"b must be greater than 4";
switch(b)
  {
case 1:cout<<"To check!";
case 1:cout<<"case number duplication!";
  }
```

And the corrected one by the system

```
#include<iostream.h>
void main(){
intsum,b;
int a[]={1,2,3,4,5};
cin>>b;
cout<<"hello World";
cout<<"it is to check";
cout<<"what can be the problem?";
cout<<"hello World";
//cout<<"Hello World";
cout<<sum;
if(b>4)
cout<<"b must be greater than 4";
switch(b)
  {
case 1:cout<<"To check!";
case 2:cout<<"case number duplication";
  }
```

AppendixVII

Sample codes of the kb-das

```
#include<iostream.h>

#include<windows.h>

#include<conio.h>

#include<stdio.h>

#include<string.h>

Void display();

Void re_check();

void main(){

        int choice;

        charans;

        cout<<"\t^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n";

        cout<<"\t\t^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n";

        cout<<"\t\t\tWELCOME TO DEBUGGING ASSISTANCE SYSTEM\n\n";

//cout<<"REMIND: remember that the compiler highlights below for an error which is above!\n";

        cout<<"\t\t&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&\n";

        cout<<"\t^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n";

        cout<<"\tNOTE:-This knowledge based debugging assistance system is a system which \thelps ";

        cout<<"novice programmers to find and correct errors while writing any \t\tprogram in turbo C++. It is developed by Tariku Fetene.";

        //cout<<"\tfullfilment of the requirement for masters degree in information science\n";

        cout<<"\n\n  *** Remind: \" ERRORS are happened on the top of the highlighted line\" ***\n ";

        cout<<"_______________________________________________________________________________\n";

        cout<<"\tWhat error message did you find? please choose from the list below.\n";

        cout<<"\t1. Undefined Symbol\t\t11. Statement missing\n";

        cout<<"\t2. Declaration Syntax error\t12. Declaration terminated incorrectly\n";

        cout<<"\t3. Unable to open include file\t13. Unterminated string or char\n";

        cout<<"\t4. Unknown preprocessor direct\t14. For statement missing\n";

        cout<<"\t5. Unexpected }\t\t\t15. Can not convert char to int\n";

        cout<<"\t6. No file name ending\t\t16. Possibly incorrect assignment\n";

        cout<<"\t7. Bad file name format\t\t17. Too many types in declaration\n";
        cout<<"\t8.   Compound   statement
missing\t18. Too many initializers\n";
        cout<<"\t9.                    Illeg
al structure operation\t19. Size is unknown\n";

        cout<<"\t10.Expression          syntax
error\t20. Unhandled exception\n";

        cout<<"\nWhat is your choice?\n";

        cin>>choice;

        switch(choice)

                {

                case   1:  cout<<"Have  you
include          library          like          this
'#include<iostream.h>'? Y or N\n";


        cin>>ans;


        if(ans=='y'||ans=='Y')


        {
        cout<<"Any  keyword  or  reserved
word which is misspelled or in uppercase?Y
or N\n";


        cin>>ans;


        if(ans=='y'||ans=='Y')


                {


        cout<<"\n\t***  Use  correct  spelling
for  keywords  and  do  not  use  uppercase!
***\n\n";


                }



        else
                {
                cout<<"What  about  missed
double quote in 'cout' statement? Y or N\n";


        cin>>ans;


        if(ans=='y'||ans=='Y')


        {


        cout<<"\n\t***  Do  not  miss  double
quote in 'cout' statement! ***\n\n";


        }
                        else


                                {


                        cout<<"Is   there   any
variable which is not declared? Y or N\n";


                                cin>>ans;



        if(ans=='y'||ans=='Y')



                {



        cout<<"\n\t***    Variable    must    be
declared before use it! ***\n\n";


                                }


                        else


                        {
```

```
        cout<<"Any missed brace in if or
loop statement? Y or N\n";

                cin>>ans;

                if(ans=='y'||ans=='Y')
                        {

                cout<<"\n\t*** Do not miss
brace for IF or loop statements! ***\n\n";

                                }

                        else

                                {

                        cout<<"did you
spelled 'endl' in wrong way? Y or N\n";


        cin>>ans;
if(ans=='y'||ans=='Y')

        {
        cout<<"\n\t\t\t***   Write   'endl'
correctly! ***\n\n";
        }
         else
                {


        cout<<"\n\tSorry! This is a new case.
No suggestion for now!\n\n";

                                }

                        }

                }
        }
        }
```

```
        }
else


        {


        cout<<"\n\t*** You have to correctly
include library first! ***\n\n";


        }

        break;

case 2: cout<<"Have you missed '#' symbol
in include library or misspelled it? Y or N\n";


        cin>>ans;


        if(ans=='n'||ans=='N')


        {


        cout<<"What about missed brace for
main function?Y or N\n";


        cin>>ans;


        if(ans=='y'||ans=='Y')


                {


                        cout<<"\n\t***    Use
correct syntax for main functions like 'void
main()' ***\n\n";


                }
```

DECLARATION

This thesis is my original work and has not been submitted as a partial requirement for a degree of master in any other university.

Tariku Fetene Sewunet, June 2017

This thesis has been submitted for examination with my approval as a university advisor.

Million Meshesha (PHD), Principal Advisor___________________

Miniychil Belay (MSc), Co-Advisor __________________________