

Building an Algorithm for Measuring C Programming Assignments Quality

by

Muhd Zulhafriz Aadel bin Marzuki

Final Report submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Business Information System)

JANUARY 2008

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Building an Algorithm for Measuring C Programming Assignments Quality

By

Muhd Zulhafriz Aadel bin Marzuki

A project dissertation submitted to the
Computer and Information Sciences Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(BUSINESS INFORMATION SYSTEM)

Approved by,

(Norshuhani bt Zamin)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

July 2008

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

MUHD ZULHAFRIZ AADEL BIN MARZUKI

ABSTRACT

The focuses of this final report is to provide a clear idea on the development of programming assignment's quality measurement algorithm. The main content of the report is the discussion of the most possible methods used to measure programming assignments quality and how it is related to this project. The proposed algorithm was modified from an existing approach used to measure software quality known as the cyclomatic measure [6]. Modifications were based on observation, studies and data collected from lecturers who currently teaches programming courses in Universiti Teknologi Petronas (UTP). The tested programming language is C considering the problems in grading programming assignments are directly related with this subject in UTP.

TABLE OF CONTENTS

CONTENTS	PAGE
ABSTRACT	ii
CHAPTER 1: INTRODUCTION	
1.0 Background of Studies	1
1.1 Problem Statement	2
1.2 Scope of Study	2
CHAPTER 2: LITERATURE REVIEW	
2.1 Software Quality Assessment	4
2.2 Measuring Programming Assignments' Quality	5
CHAPTER 3: MEHODOLOGY	
3.1 Research Methodology and Project Activities	6
3.2 Programming Assignments' Quality Measurement Algorithm Flow chart	7
3.3 Testing Method Flow Chart	9
3.4 Tools	11
3.5 Screen Designs	12
CHAPTER 4: RESULTS AND DISCUSSION	
4.1 Data Gathering and Analysis	14
4.2 Functionality Analysis	22
CHAPTER 5: CONCLUSION AND RECOMMENDATION	28

REFERENCES	29
APPENDICES	31
Appendix A- Gantt Chart	31
Appendix B- Programming Assignments' Quality Measurement Criterion	32
Appendix C- Algorithm for Measuring C Programming Assignments Quality Soft Codes	37

LISTS OF FIGURES

Figure 3.20: Programming Assignments' Quality Measurement Algorithm	7
Flow chart	
Figure 3.21: Different structure of sentences with same meaning	8
Figure 3.22: Example of lowercase and uppercase letters	8
Figure 3.30: Testing Method Flow Chart	10
Figure 3.40: VB.Net Logo	11
Figure 3.50: Programming Assignments Quality Checking Algorithm	12
Figure 4.10: Number of Variables	15
Figure 4.11: Data Types	16
Figure 4.12: Relevant Comments	17
Figure 4.13: Program Neatness	17
Figure 4.14: Output Generated	18
Figure 4.15: Logic	18
Figure 4.16: Syntax Correctness	19
Figure 4.17: Criterion's Importance Value on Percentage	20
Figure 4.20: Prompting Student's Answer Selection	22
Figure 4.21: Student's Answer Selection	23
Figure 4.22: Prompting Answer Scheme Selection	24
Figure 4.23: Answer Scheme's Selection	25
Figure 4.24: Preprocessed Students Answer & Answer Scheme	26
Figure 4.25: Analysis Screen	27

LIST OF TABLE

Table 4.1: Programming Analysis Criterion and Their weighted value (%)	15
--	----

CHAPTER 1

INTRODUCTION

1.0 Background of Studies

In learning programming, students must undergo intensive drills in writing codes to improve their familiarity with the syntax. However, these training must also be evaluated continuously to assess the standard of students and rectify how they can improve their code writing abilities. To achieve precise and reliable assessment method, a standard accepted criterion must be identified to be applied in the assessment process. Each semester, the number of students enrolled for Structured Programming subject had increased massively from 100 to more than 600 students at one time [15]. This is due to the university requirement which oblige all students to undertake Structured Programming subject during their first year of studies in UTP. C language as one of structure programming languages was thought for Structured Programming subject. The syllabus of this course will be covering basic programming structure focusing on strengthening the foundation of programming among students. These basic structures would be covering the looping control structure, selection control structure, function, dynamic data structures and static data structures.

The main objective of this project is to identify the most appropriate software quality measurement algorithm that can be dynamically applied to any types of programming assignments regardless of the language. Unlike previous research and findings which focuses on the applicability of the metrics only to large and medium size software (10^3 or more line-of-codes), this project aims to provide sufficient measure for programming assignment which relatively smaller than normal software ($10^1 - 10^2$) apart from identifying criteria required by the assessor but not considered important in measuring software quality. This special requirement is crucial due to the concern of training students for good programming practice. In order to do so, a general accepted criterion will be collected from a group of lecturers in UTP as the sample of the research based on the criterion suggested by Ravindran [13]. He quoted that

attributes such as reliability, robust, reader friendly, readable, conforming to standard and proper documentation are essential for a good programming codes.

1.2 Problem Statement

With vast amount of students enrolling for Structured Programming course each semesters, there had been some concerns over the time constraint faced by lecturers to assess and grade the assignments given to students. Since the ratio of lecturers to student for the Structured Programming subject is relatively high, each lecturers are burdened with a lot of assignments need to be marked. To make things worst, they have very limited time to complete the assignments' marking. It is very important for lecturers to marks all the assignments within the timeline as students need feedback on their weakness and further improves their skills.

Finally, as lecturers are burdened with limited timeline with vast amount of assignments, they are prone to human error factors. It is very likely that the quality of marking will be deteriorated as more assignments were marked. Among the possible human error factors occurring is biasness between assignments as the manual marking is obviously not conducted anonymously.

For the past decades, several softwares had been developed to evaluate and identify the correctness of programming assignments. These programs, such as HoGG [1], ASSYST [2] and CAP [3], applied dynamic analysis approach which compares any software's output with the expected output. In recent years, there has been software measuring both the correctness and the quality of the software known as AutoGrader [4] and Automated Marking System (AMS) [5] which provides real-time assessment. Even though these softwares introduced improved features of marking assignments, however, to date there is no software that was built to measure the quality of programming assignments apart from the correctness.

1.3 Scope of Studies

Before the algorithm is constructed, researches will be conducted to determine if there is similar project was done before and whether or not it can be referred to in completing this project. Another factor that needs to be examined is the suitability of the criteria suggested- if there is

any. One alternative that is suggested should there be no appropriate modules applicable directly to this project is to develop our own criteria by gathering data through questionnaire with the lecturers from Computer and CIS department as a sample.

These suggested modules were however, focuses on measuring quality of large software (10^4 line-of-codes). Since programming assignments were usually consisting of small sized program ($10^1 - 10^2$ line-of-codes), normal software quality metrics were insufficient for marks awarding. Hybrid approach will be identified and some modification may prove to be helpful in constructing this algorithm. Researches will be carried out to identify the criterion stressed by lecturers instead of models suggested such as Cyclomatic Measures, Lines-of-Codes (LOCpro) and Volume.

On another aspect, the scope of this project is to implement a static analysis approach to evaluate and assess students' programming assignments. Thus, some criterion which is of high importance may not be covered in this projects considering implementing dynamic analysis approach for C would require C language compiler and ways to link this compiler to this project.

CHAPTER 2

LITERATURE REVIEW

2.1 Software Quality Assessment

Since 1976, they have been many researches working on finding the best criteria to assess the software quality accurately. Basically, the criterion suggested can be classified into several groups- Complexity [6], [7], [8], Operational Reliability [9], Endurance Test Time [10] and Error Rates and Densities. Based on these suggested groups, it was found that, numerous were done focusing on software's complexity as the accepted approach to measure software quality. Referring to the research done by Bowen [11], he had concluded that the complexity measure has the highest sensitivity to the intra module's interaction. On further research findings by Takahashi [12], by doing some modification on the formula $S_1 = e - n + 2$, this approach could be applied to the smallest unit in C programming- corresponding to a function in C, to the inter-module level of a system.

However, while this approach does measure a software quality accurately, it does not provides sufficient and appropriate guidelines that can help achieve the goals of training students to good programming practices. According to Ravindran [13], a good program, should exhibit these attributes apart from correctness; reliable, robust, user-friendly, efficient, readable and portable. Software should always function correctly, regardless over a period of time, or over ranges of different data. Robustness deals with the ability to detect inappropriate data and handle them properly. Other than the ease of use, it must achieve the desired result with the best way and shortest time span.

2.2 Programming Assignments' Quality Assessment

Several software are available to automatically mark the programming assignments of various languages. Among the latest software is AMS [5] and AutoGrader [4] while some of the earliest software are HoGG [1], ASSYST [2] and CAP [3]. However, this software were found did not identify the quality of an assignment objectively and before awarding appropriate marks to a particular student. An approach applied by these systems is called the dynamic analysis approach, which executes programs built on a real or virtual processor [16]. Special libraries are required for this analysis method and for some cases; recompilation may be needed [16].

In 2006, the Web-based Automated Grading System (WAGS) was developed to automatically grade programming assignments written in C, VB.Net and Java [15]. Using static analysis, this system compares the students' codes with the answer scheme. A feature highlighted in this software is the ability to accept multiple answer schemes with the assumption of student's answer is unique and distinctive, thus multiple schemes are needed. Static analysis however, does not require any special libraries thus; software evaluating was done by examining the codes line-by-line.

Based on the attributes coined by Ravindran [13], some modification will be made together with the elaboration on the important aspects focused by lecturers in awarding marks to students. An algorithm is proposed to be developed which works as an expansion patch to the previously developed WAGS [15] system.

CHAPTER 3

METHODOLOGY

3.1 Research Methodology and Project Activities

The methodology used for this development of this algorithm is known as prototyping [17]. Considering the small size of this project and the advantages provided by this method which enables flexibility in algorithm development, it is believe that this approach could assist within shorter period of project's development.

Questionnaires were distributed to the lecturers to gather the criterion needed and deemed most appropriate in assessing students' programming assignments apart from emphasizing good programming practices. The outcome of the questionnaires was analyzed before assigning certain weight to each and every criterion met in the assignments according to the level of importance accordingly.

For each of the criterion, the marks awarded will have different values. The weight of each mark will be decided after analyzing the result of the questionnaire. The objectives is ensure that the most crucial criteria can be stressed by giving higher value of marks while the less important criteria plays lesser roles in contributing more marks for students.

3.2 The Algorithm Flow chart

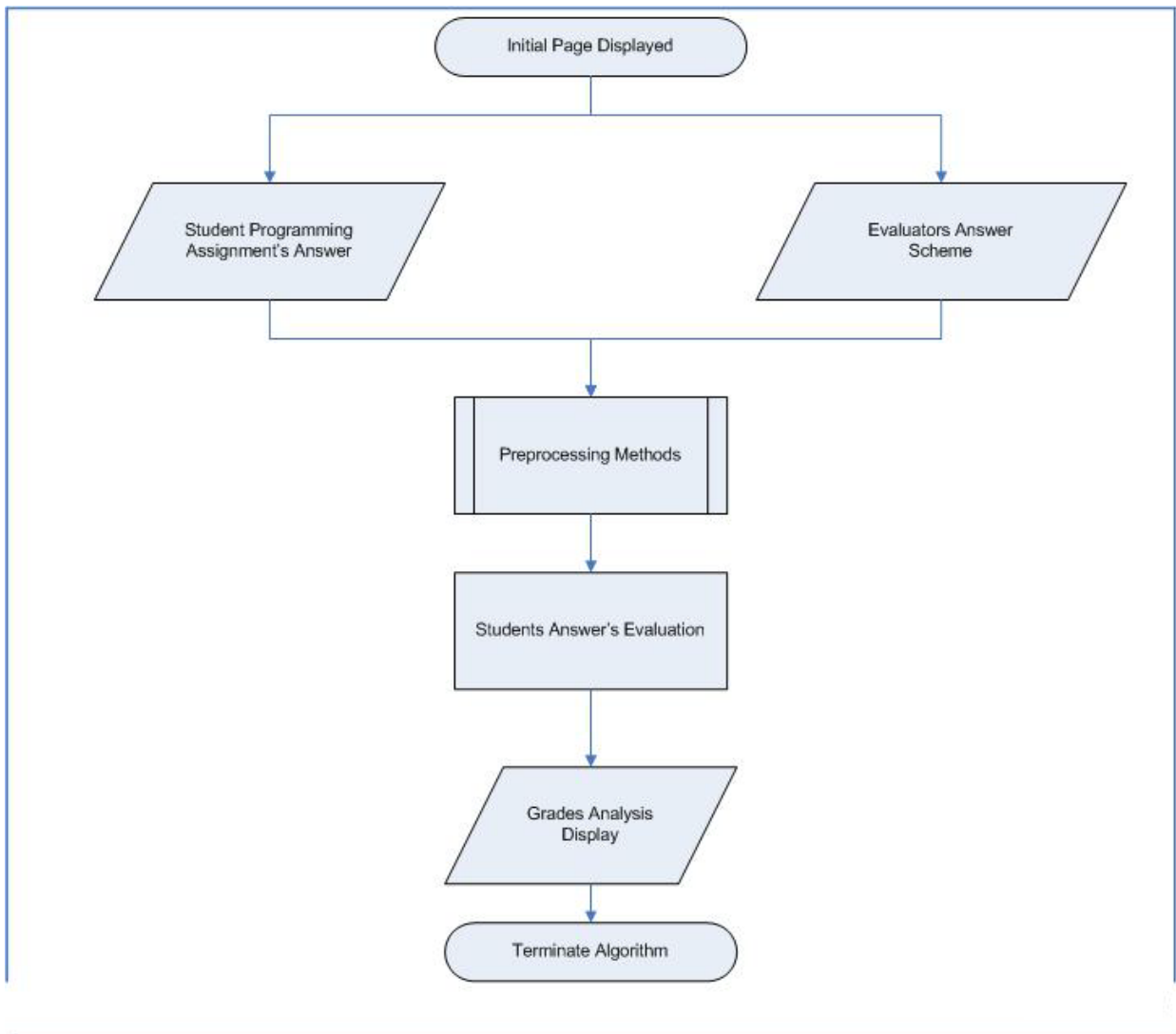


Figure 3.20: Programming Assignments' Quality Measurement Algorithm Flow

The flowchart describes the flow of the algorithm. Every line of codes will be read and check for the selected criteria. Appropriate marks will be awarded to the students if any of the criterions was met and vice versa. The same process of evaluation will be repeated for every line until the end of the programming assignments. Before displaying the total marks, all marks for each criterion will be sum-up including the marks awarded for the correctness aspect generated from WAGS [15].

The approach taken in completing this project is to work on each individual module (function). Marks for the selected criterion (Table 4.10) will be generated individually by each module. Following that, each of the marks generated by these individual modules will be weighted accordingly on their importance before obtaining the final marks. By grading based on separated module, not only it will ease the development of the algorithm, it will also help on displaying a student's weaknesses and strength for each of the criteria. In addition to that, details portion of marks earned for each criteria can be displayed separately.

A suggested by Norshuhani[15], a line-by-line comparison between students' answer script and the answer scheme provided by the lecturer are required in order to determine the correctness of students' answer. However, problems arise when there is difference in strings written by users and evaluators which is not exactly an error (figure 3.21). In addition to that, a system would not recognize the same letter with uppercase or lowercase (figure 3.22). Several other issues of concern involved the difference in name of variables declared for each answer- difference of variables name is not an error, but the system will read it as error with the comparison approach, and

```
Printf("\nEnter two numbers : "); /* The first method of asking for 2 input*/  
Printf("\nPlease Key in 2 numbers : "); /*Another method of asking for 2 input*/  
/*Both method is the same but with different sentences structure*/
```

Figure 3.21: Different structure of sentences with same meaning

```
#include<stdio.h> /* Same sentences which are unrecognizable by system due to*/  
#INCLUDE<STDIO.H> /*the lowercase and the uppercase of letters*/
```

Figure 3.22: Example of lowercase and uppercase letters

In order to ensure that the grading process will run efficiently apart from minimizing the errors while grading, basic preprocessing will be made before each grading process. Referring from WAGS system [15], preprocessing functions were modified to include several respective processes:

- a) Codes submitted will be converted to lower case [15].
- b) All empty lines in between codes are to be removed [15].
- c) Empty spaces at the beginning and starting of each line are to be trimmed while codes are to be left justified [15].
- d) In order to shorten the time consumed grading the codes, a standard text will replace all system generated codes [15].
- e) Any programmers define codes will be replaced with a standard text as following:
 - a. VAR will be used to all variables name declared within the codes [15].
 - b. FREETEXT will be used to any text generated in the interface [15].

3.3 Testing Method Flow Chart

In verifying the result generated by this algorithm to evaluate its reliability, some analysis and comparison will be made. Conventional grading approaches by one of the lecturers are to be compared with the grades awarded by applying the algorithm before both results' reliability will be determined by referring to the criterion gathered through questionnaire earlier. The diagram presented will gives visual explanation on the test method.

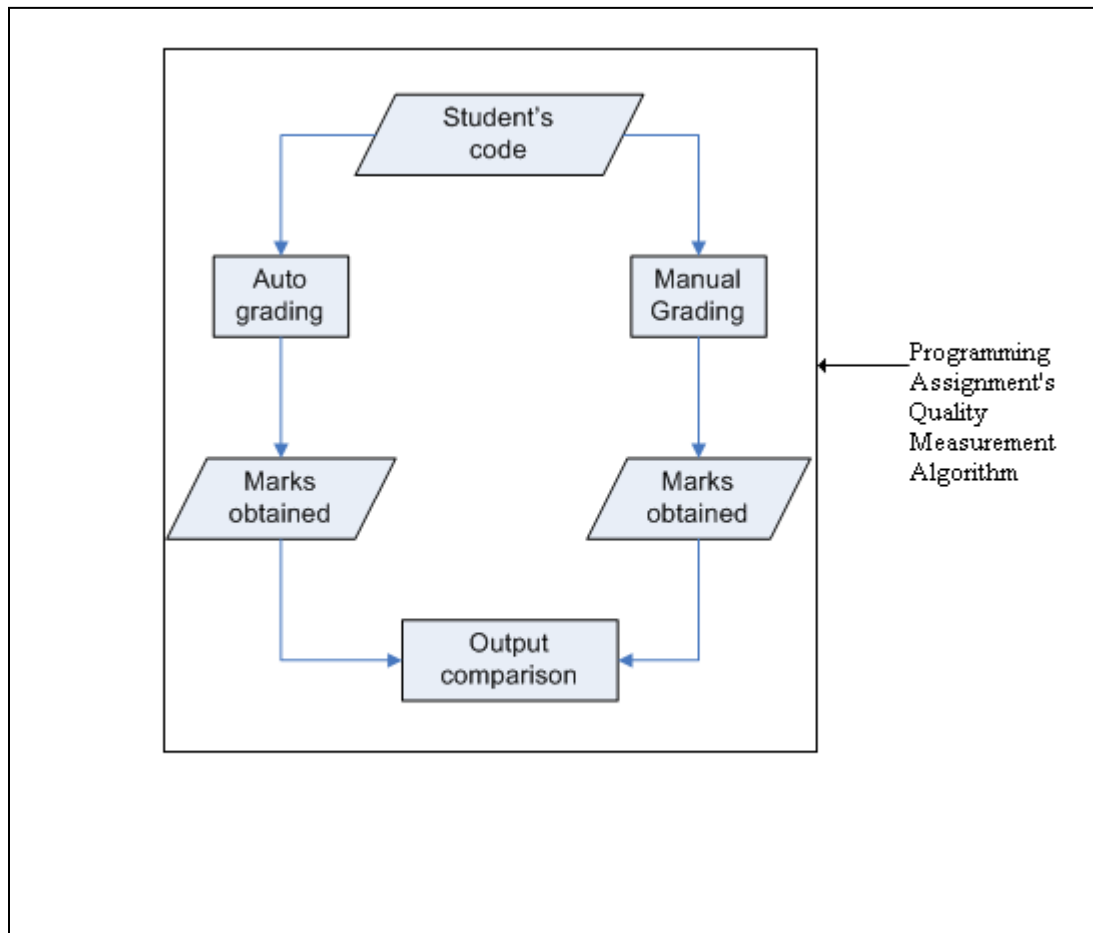


Figure 3.30: Testing Method Flow Chart

On the aspect of consistency, a stress test will be conducted to get an output compared between the manual grading system and automated grading system. Dispersion between marks will be calculated to measure the consistency of marking for both approaches (manual and automated). 50 answer scripts will be assigned to a selected assessor while at the same time, the same answer scripts will be assessed with the system. Smaller dispersion value shows higher consistency while higher dispersion indicates higher inconsistency due to various reasons.

3.4 Tools

- Microsoft VB.NET



Figure 3.40: VB.Net Logo

VB.Net was chosen as the platform of this project. As mentioned in several parts of this reports, the scope of this project is to compliment WAGS [15] apart from focusing on evaluating the quality of students' programming assignments. Since WAGS [15] was also implemented in VB.Net and it was proven a success, it is preferable that the same language will be used to prevent any issue of compatibility and integration between different languages.

3.5 Screen Designs

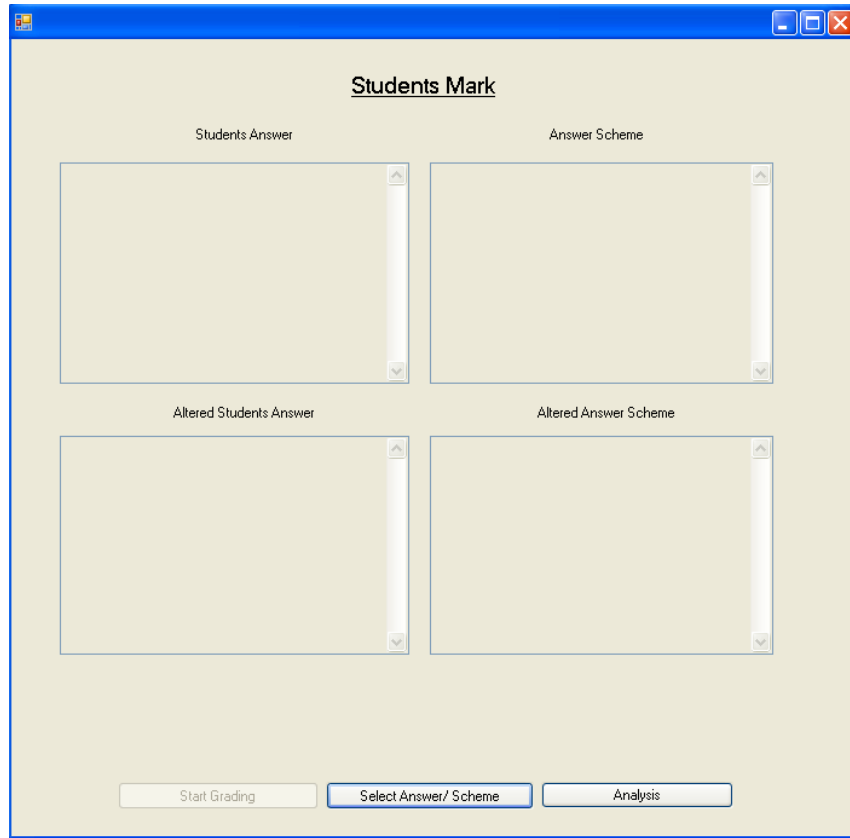


Figure 3.50: Programming Assignments Quality Checking Algorithm

The focus of this project is not the development of a system to evaluate students programming assignment. Instead, it is to create an algorithm that can cater the goals and objectives as described earlier. Thus, the development of this screen is solely to assist in monitoring the success of the said algorithm. As shown by the figure 3.50, the top left side of screen would be displaying students answer before preprocessing activities had taken place. On the top right side, is where the answer scheme uploaded by the evaluator will be displayed. On the lower left part and lower right side of the screen, the preprocessed students answer and answer scheme will be displayed each accordingly.

On the lowest part of the screen, there will be 3 buttons that serve different function respectively. The middle button is used to select both students answer and answer scheme. On click event of this button, the preprocessing functions will be automatically executed before the preprocessed answer will be displayed. User can compare the output of each answer after being preprocessed before proceeding to the next function.

The start grading button will only be enabled after user is done with selecting both students answer, answer scheme and preprocessing the codes. This button will execute the evaluating process which includes line by line comparison and the quality checking. Following that, the analysis button will be enabled to aid user in getting detail analysis on their weakness and strength for each of the criteria evaluated. This way, proper action can be taken to improvise on their weakness while maintaining their strength.

It is very important to take note that the design of the buttons may be redundant. With extended time line, this buttons are possible to be made lesser than it is now up to only 1 or 2 buttons only. However, as said earlier, the development of the screen would only serve the need to prove that the algorithm built is functioning properly with addition to some test that will compliment each other.

CHAPTER 4

RESULTS AND DISCUSSION

In order to obtain precise and detail marks which will be awarded to specific students, some value of each criterion adhered to need to be decided. To ensure that the value given is accepted by assessors, a questionnaire was designed to grasp some ideas on the important criteria and its most appropriate value.

4.1 Data Gathering and Analysis

10 out of 18 questionnaires were collected from different lecturers. These questionnaires were made of 8 questions measuring different aspects of quality. Each of these questions is required to be assigned with importance value ranging from 1-5 with one being the least importance and 5 being of utmost importance. An analysis will be done for each of these questions and an average value will determine the weight of each quality. Together with this criterion, we will add up to the marks awarded by WAGS system. The criterion covered in the questionnaire is as follows:

Input Checking

Input checking deals with the ability to intercept and stop program from running if invalid data types were entered by user. E.g. appropriate error handler

$$\begin{aligned}\text{Importance Level}_{\text{Input Checking}} &= \frac{4 + 5 + 5 + 5 + 4 + 5 + 5 + 5 + 3 + 4}{10} \\ &= 4.5\end{aligned}$$

Number of Variables

This deals with the number of variables used to derive to an answer (a student may use 4 variables (Figure 4.10(a)) while other may use 3 variables (Figure 4.10(b)) to obtain the same result). Fundamentally, fewer variables used the speed of the program increases.

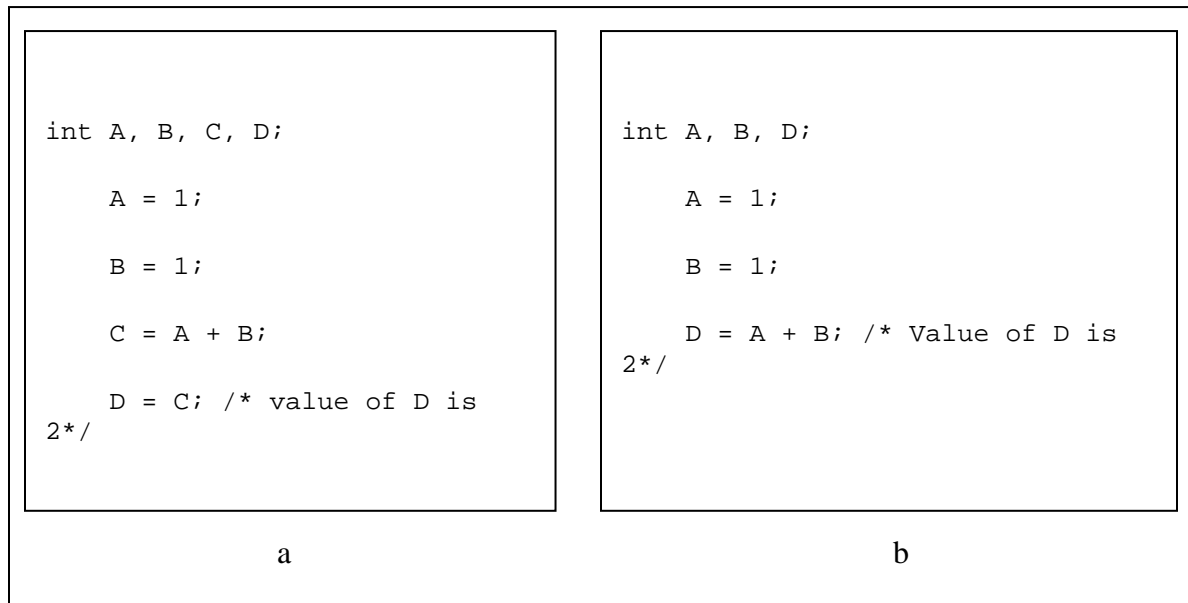


Figure 4.10: Number of Variables

$$\begin{aligned} \text{Importance Value Variables amount} &= \frac{3 + 1 + 4 + 4 + 1 + 3 + 4 + 2 + 5 + 4}{10} \\ &= 3.1 \end{aligned}$$

Data Types

This emphasized the usage of specific data type used. E.g. Float or Double data type instead of integer (The data type used should be able to hold the size of particular values (Figure 4.12(b))).

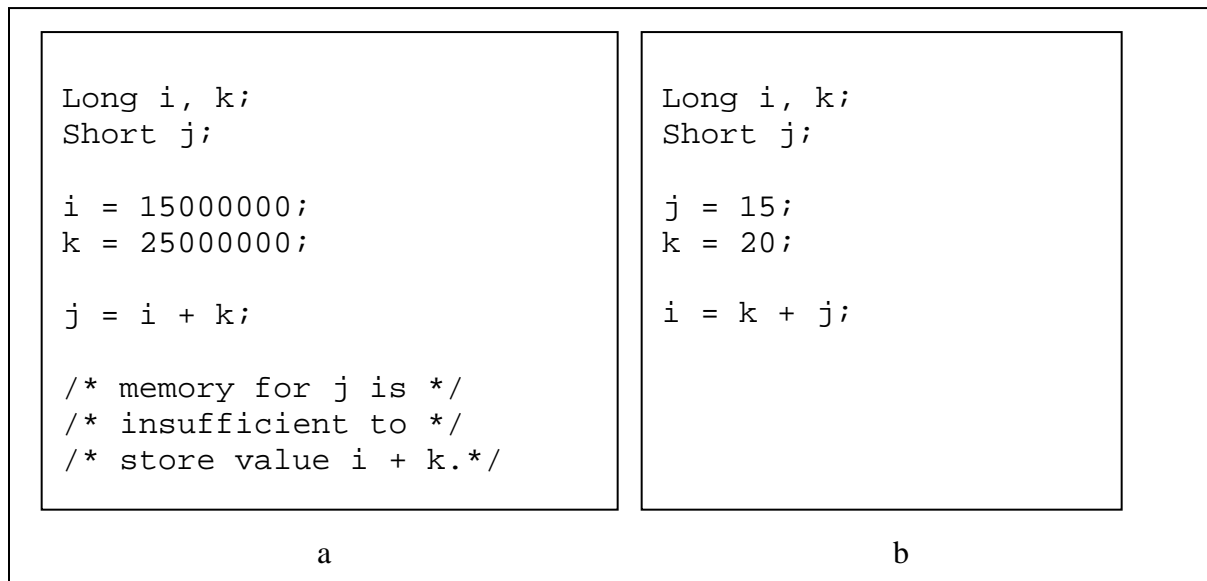


Figure 4.11: Data Types

$$\begin{aligned}\text{Importance Value}_{\text{Data Types}} &= \frac{5 + 5 + 4 + 3 + 3 + 5 + 3 + 4 + 4 + 4}{10} \\ &= 4.0\end{aligned}$$

Relevant Comments

Whether the codes built by students comes with a comments that describes their program briefly and its relevance (Figure 4.12) to the program.

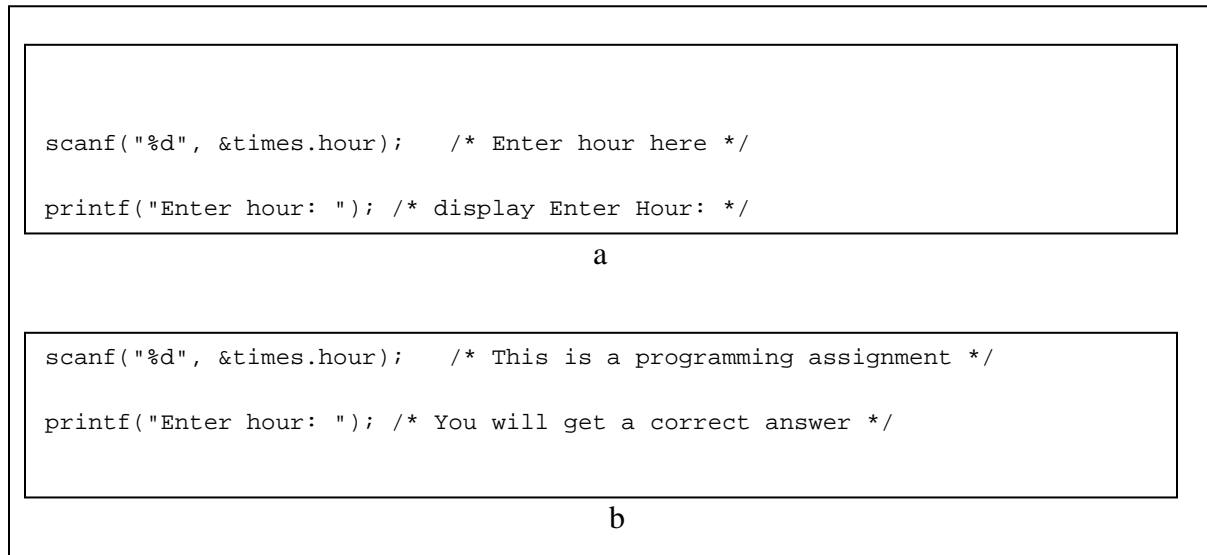


Figure 4.12: Relevant Comments

$$\begin{aligned}\text{Importance Value}_{\text{Comments}} &= \frac{5 + 5 + 3 + 2 + 2 + 4 + 4 + 5 + 4 + 5}{10} \\ &= 3.9\end{aligned}$$

Program Neatness

Does the particular student starts the next line properly (Figure 4.13(a)) or lump all the codes into one line (Figure 4.13(b))?

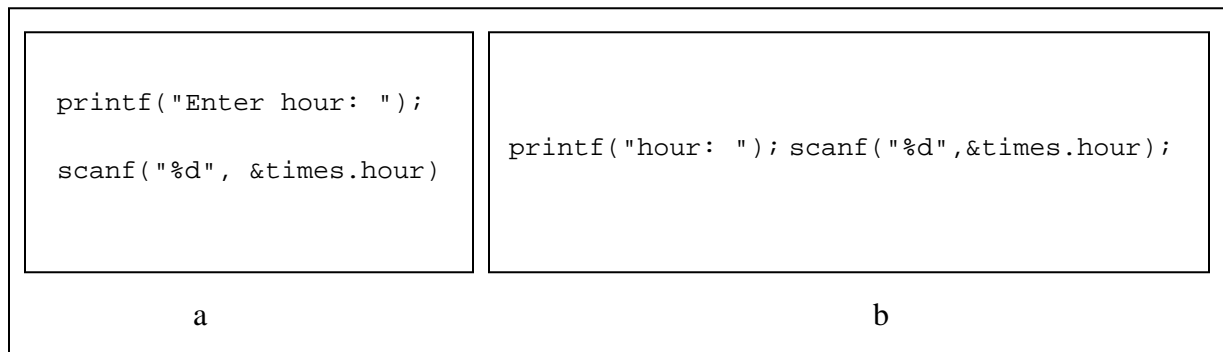


Figure 4.13: Program Neatness

$$\begin{aligned}\text{Importance Value}_{\text{Program Neatness}} &= \frac{3 + 3 + 3 + 3 + 5 + 5 + 5 + 2 + 5 + 5}{10} \\ &= 3.9\end{aligned}$$

Correctness- Output Generated

This aspect deals with the ability to generate the expected output (Obtained the correct outcome logically (Figure 4.14)).

```
int a, b, c;
printf("/nEnter two numbers : ");
scanf("%d%d",&a,&b);
c = a + b;
printf("/nNow sum is now %d", &c);
```

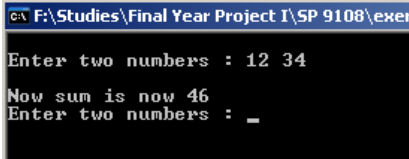


Figure 4.14: Output Generated

$$\begin{aligned} \text{Importance Value}_{\text{Output Generated}} &= \frac{5 + 5 + 5 + 5 + 5 + 5 + 5 + 5 + 5 + 4}{10} \\ &= 4.9 \end{aligned}$$

Correctness - Logic

Logic criteria deal with the structure and output of a loop. The loop structure can be executed, but it doesn't have upper limit causing it to loop endlessly (Figure 4.15).

```
for (k=2; k>0; ++k)/*This loop would not stop looping*/
{
printf ("Inner %3d%3d/n", i,k);
}
```

Figure 4.15: Logic

$$\begin{aligned} \text{Importance Value}_{\text{Logic}} &= \frac{4 + 5 + 5 + 5 + 5 + 5 + 5 + 5 + 5 + 5}{10} \\ &= 4.9 \end{aligned}$$

Correctness- Syntax

Syntax error is basically dealing with the error generated by students while doing their assignments. Students' codes were not complying with the C language's syntax (Fig. 4.16).

```
/* The 'print' command line should be 'printf' */  
print("Inner %3d%3d/n", i,k);
```

Figure 4.16: Syntax Correctness

$$\begin{aligned}\text{Importance Value}_{\text{Syntax}} &= \frac{5 + 5 + 4 + 5 + 5 + 5 + 5 + 5 + 5 + 4}{10} \\ &= 4.8\end{aligned}$$

Table 4.1: Programming Analysis Criterion and Their weighted value (%)

		Importance Value	Importance Value (%) (X/25.6) x 100
No. of Variables Used		3.1	12.1 ≈ 12
Data Types		4.0	15.6 ≈ 16
Neatness		3.9	15.2 ≈ 15
Correctness	Output	4.9	19.1 ≈ 19
	Logic	4.9	19.1 ≈ 19
	Syntax	4.8	18.8 ≈ 19
Total		25.6	100

Referring to the questionnaire's analysis provided earlier on this chapter, the first criteria, input checking and the 4th criteria, relevant comments, were not considered in this project although it was considered crucial to the assessors. The development of this algorithm was intended to be that complimenting the WAGS system which was built previously. Thus, it is most important

that this algorithm is compatible to WAGS system. Since WAGS is implementing a static analysis approach to evaluate the correctness of the programming assignments, this project will be using the same method to generate any output. On the other criterion discarded, the comments criteria, the identification of relevant comments would require new system that can evaluate each comment written by students. Considering the scope of studies and its complexity, this function was suggested to be built for enhancements of this algorithm.

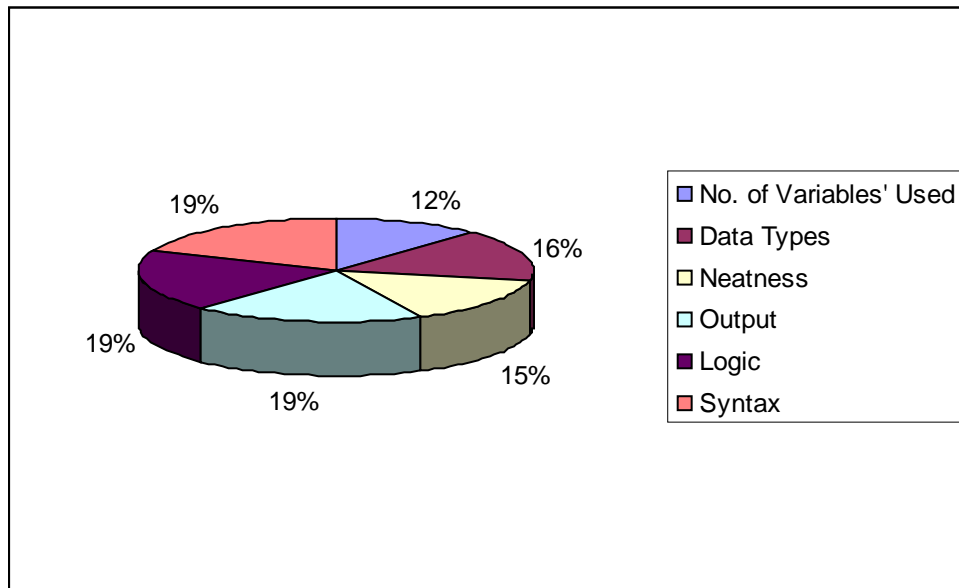


Figure 4.17: Importance level of criterion

The figure shown (Figure 4.17) is the total of the entire criterion displayed on a pie chart which consists of different criteria and their value of weight respectively. In awarding marks to the student, there some underlying assumption made. Initially, all students are allocated full marks. As the algorithm was executed, any errors detected would result in deduction of marks based on the value assigned. In turn, this approach is encouraging students to write shorter codes, that can derived to the desired output than longer and more complex codes. Shorter codes can promote higher efficiency and smaller probability or syntax error and other arising issues. From the shown pie chart, the calculation method proposed is based on the total marks of specific question. To give and easy explanation, consider the following scenario:

Total marks for each question = 20.

Marks which will be deducted for each error made:

- i. No. of Variables Used = 19% of total marks
= 0.19×20
= 3.8 marks will be deducted for each error.
- ii. Data Types = 16% of total marks
= 0.16×20
= 3.2 marks will be deducted for each error
- iii. Program neatness = 15% of total marks
= 0.15×20
= 3 marks will be deducted for each error
- iv. Output Correctness = 19% of total marks
= 0.19×20
= 3.8 marks will be deducted for each error
- v. Logic Correctness = 19% of total marks
= 0.19×20
= 3.8 marks will be deducted for each error
- vi. Syntax Correctness = 19% of total marks
= 0.19×20
= 3.8 marks will be deducted for each error

4.2 Functionality Analysis

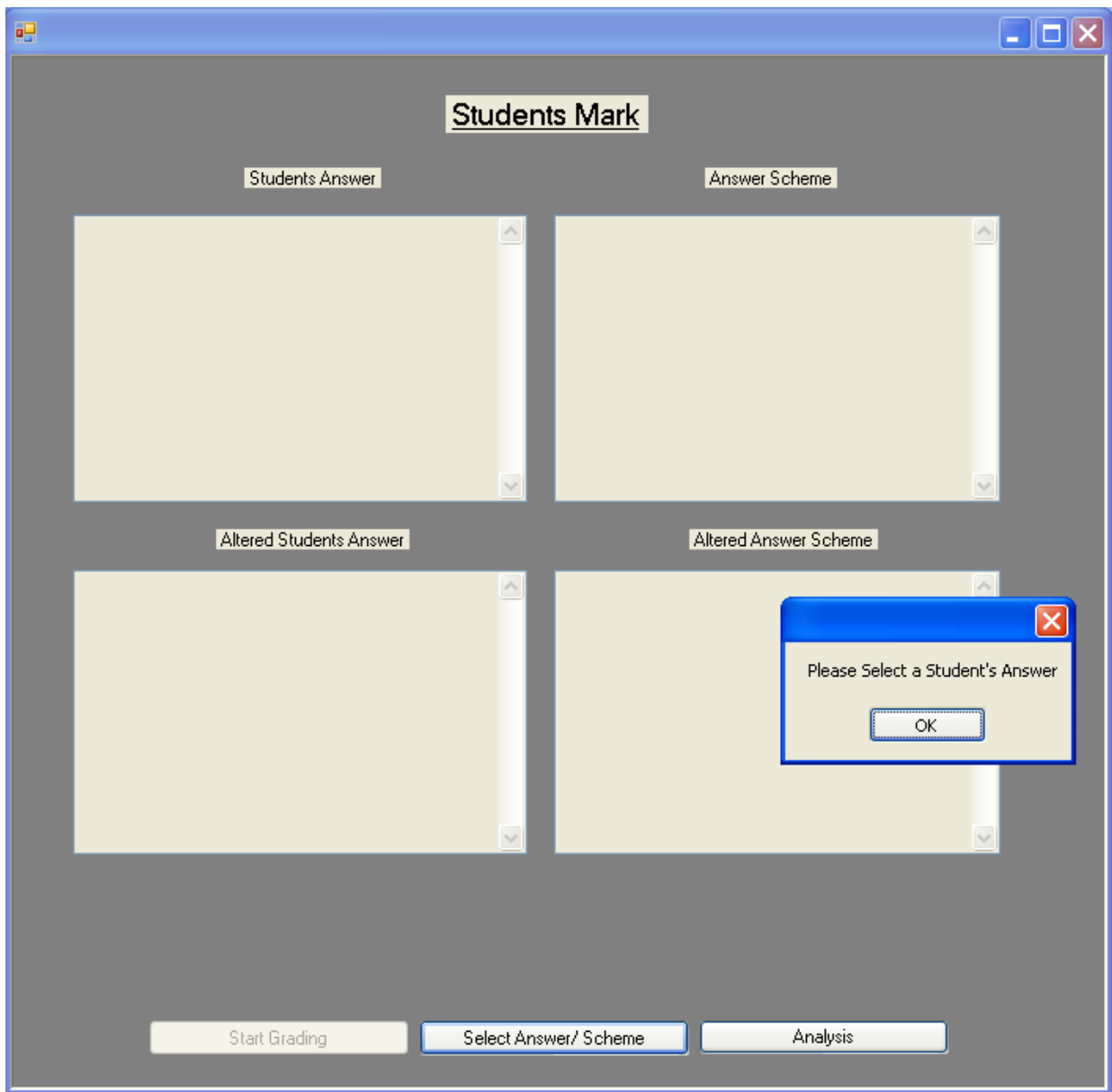


Figure 4.20: Prompting Student's Answer Selection

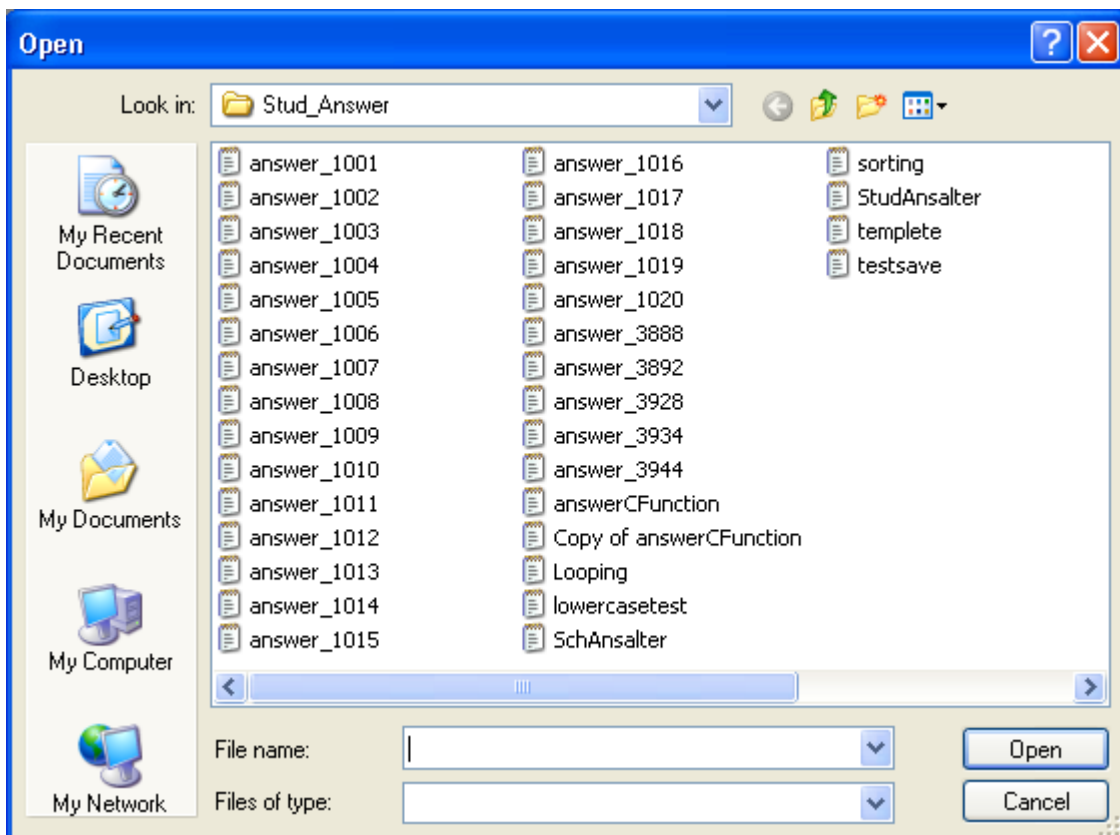


Figure 4.21: Student's Answer Selection

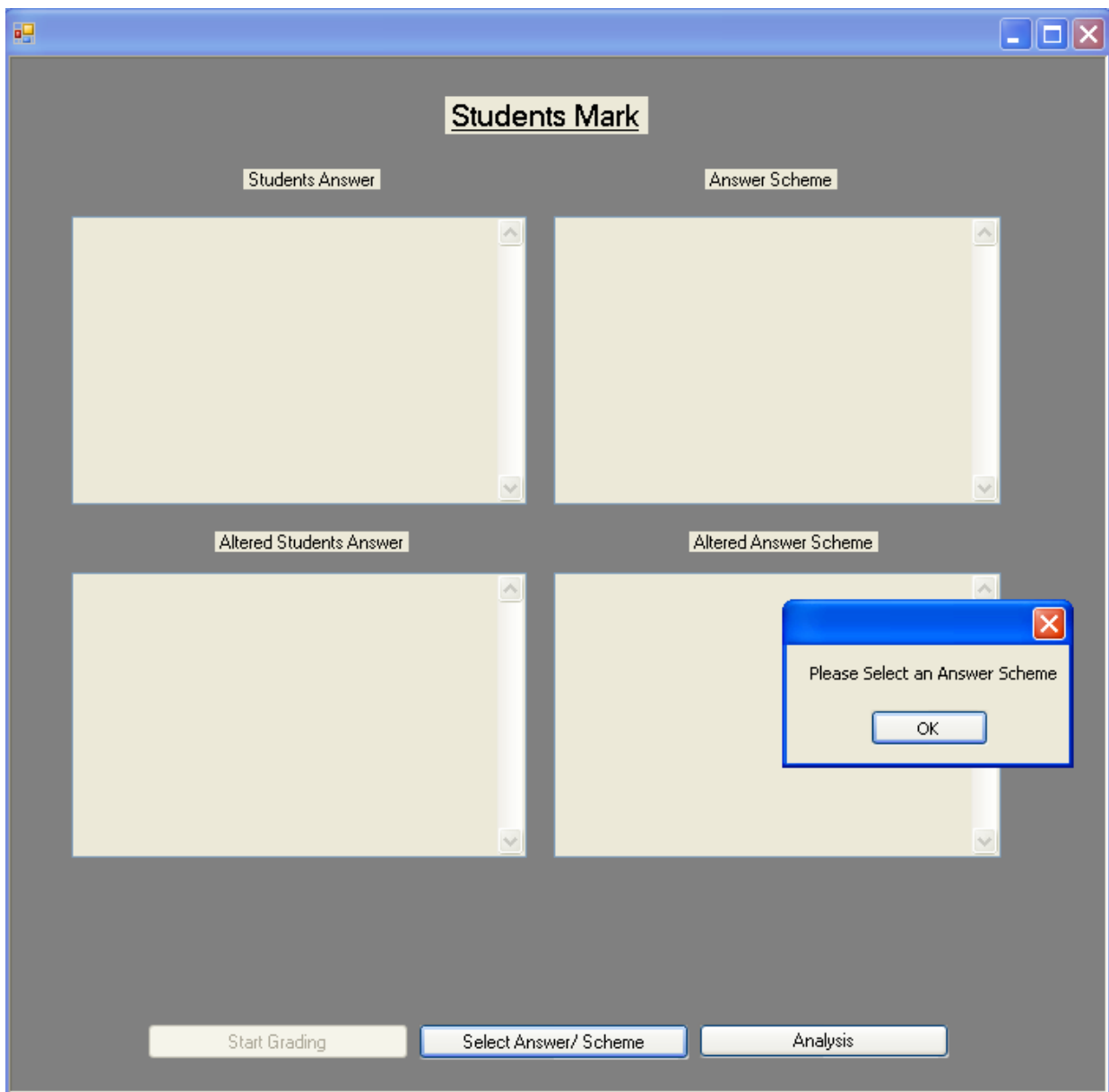


Figure 4.22: Prompting Answer Scheme Selection

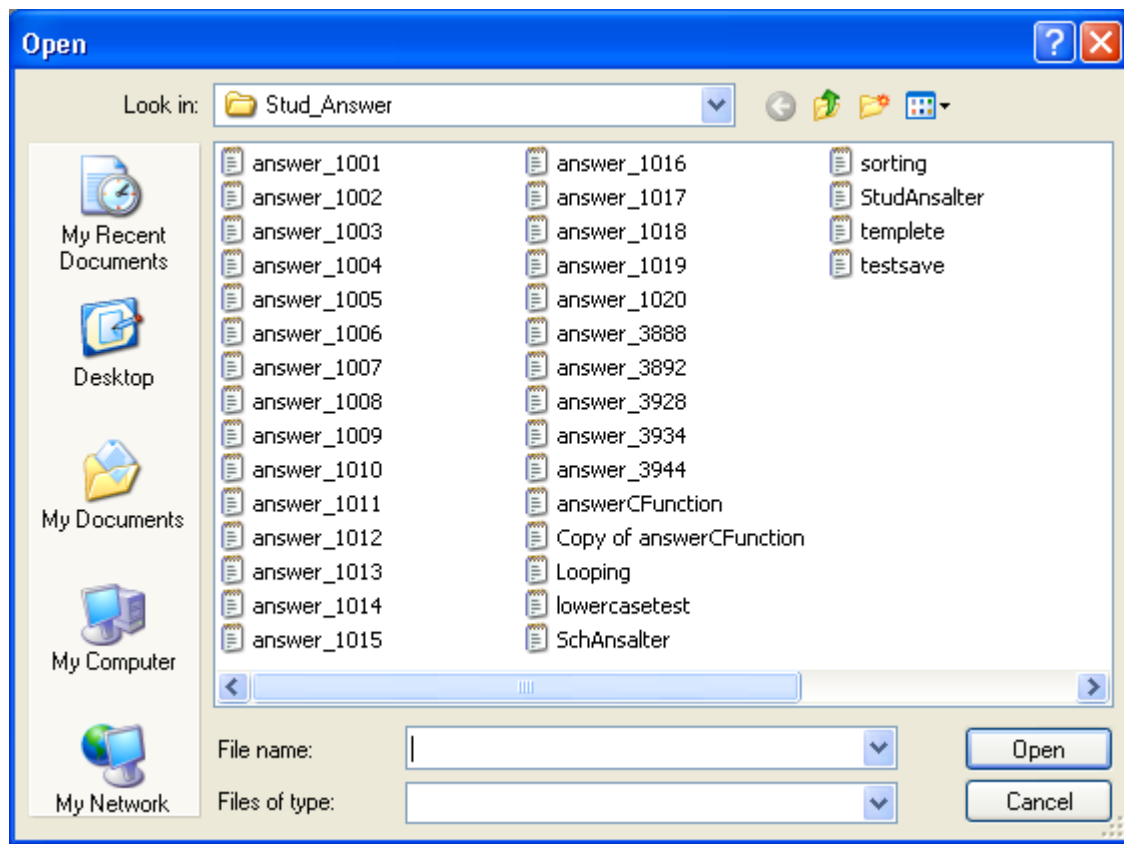


Figure 4.23: Answer Scheme's Selection

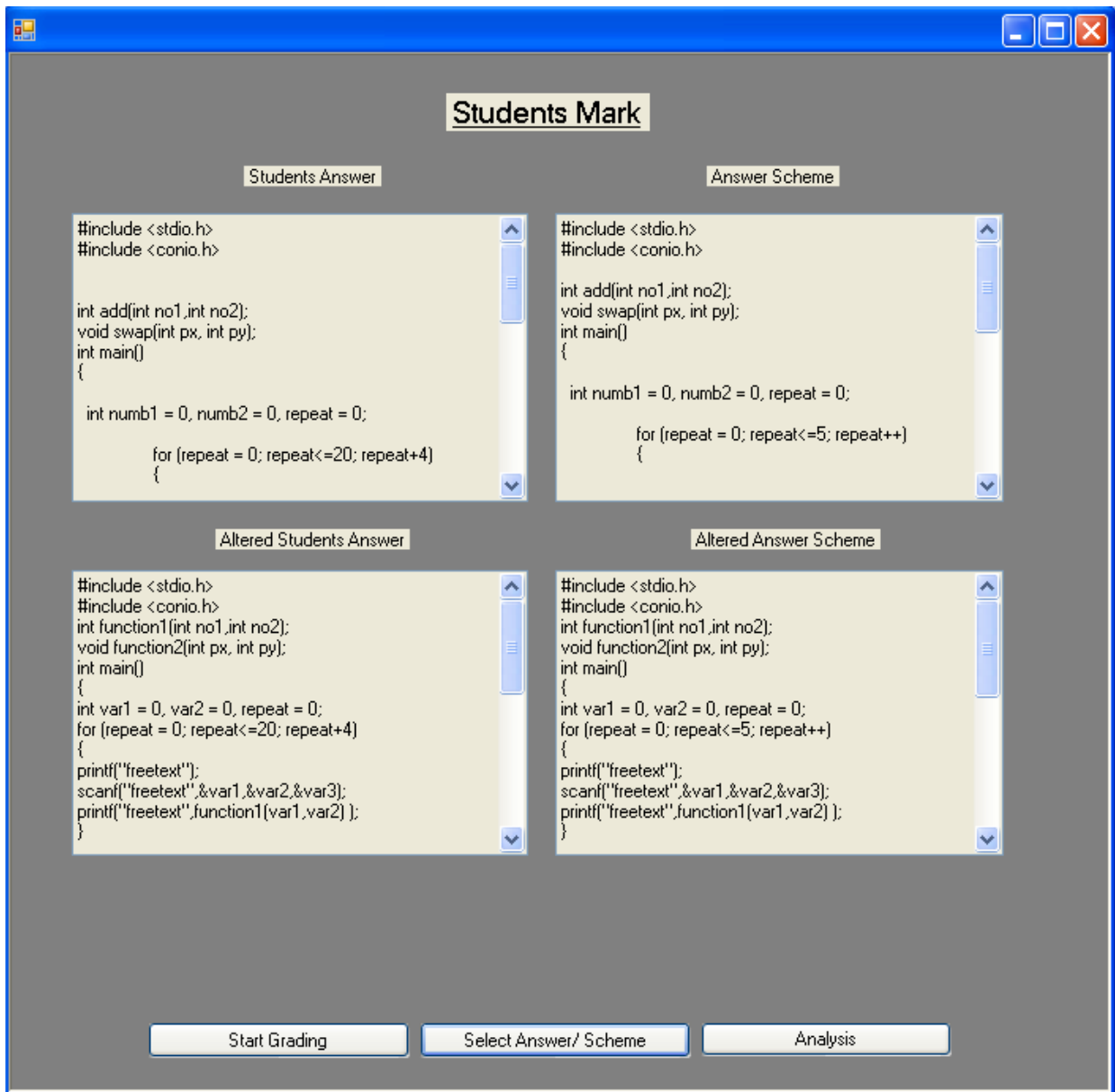


Figure 4.24: Preprocessed Students Answer & Answer Scheme

Users can initiate the grading process by simply clicking the Select Answer/Scheme button. A dialog box will appear prompting user to select a student answer (Figure 4.20). Following that, user can chose the file from the select file dialog box (Figure 4.21). The same process will be repeated for the selection of answer scheme (Figure 4.22) (Figure 4.23). Figure 4.24 indicate the output expected after the preprocessing function had been executed. Both the processed and raw answer was published at the same screen in order to ease users to compare the difference between both answers (raw and preprocessed).

The image shows a Windows-style window titled "Form1" with a blue title bar and standard minimize, maximize, and close buttons. The main area has a light beige background. At the top center, the text "Marks deducted" is displayed. Below this, there is a list of criteria on the left and corresponding numerical values in text boxes on the right. The criteria and values are: "No. of Variables Used" (6), "Data Types" (2), "Codes Neatness" (0), "Output Correctness" (0), "Logic" (0), "Syntax" (3), and "Total" (11). At the bottom center, there is an "OK" button with a dotted border.

Marks deducted	
No. of Variables Used	6
Data Types	2
Codes Neatness	0
Output Correctness	0
Logic	0
Syntax	3
Total	11

OK

Figure 4.25: Analysis Screen

Upon completion of evaluation event, analysis button will be enabled. Once the analysis button was clicked, user will be redirected to another form displaying all the portion of marks deducted for each criterion (Figure 4.21). The OK button will closed the form and redirect user to the previous screen.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

Until recent years, researchers had been studying and developing new software that can improve programming assignments grading process. Most of them had opted for dynamic analysis approach rather than static analysis approach. However, realizing the advantages of static analysis as discussed above, WAGS [15] were developed to cater the needs of evaluating programming assignments of various languages without any requirement for specific language compiler. By focusing on the quality aspect, this expansion project would enable marking process to be more detail, precise and accurate. It is hoped that this project will be a stepping stone for future projects that can shift the burden of marking programming assignments away from lecturers thus, benefiting students indirectly.

It is recommended that this project to be further enhanced and reviewed from time to time to cater the needs of the lecturers in evaluating programming assignments. Serious efforts are required in implementing this algorithm together with WAGS [15] system although some issues may arise from the implementation of this project. As time goes on, provided that this algorithm is enhanced on regular basis, it will reach its maturity stage where the algorithm is stable and reliable.

References

- [1] Morris, D.S 2003. Automatic Grading of Students' Programming Assignments: An Interactive Process and Suite of Programs. Proceedings of 33rd ASEE/IEEE Frontiers In Education Conference.
- [2] Jackson, D. 1997. A Software System for Grading Student Computer Program. Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education. 28:335-339. ACM Press
- [3] Schorsch, T. 1995. CAP: An Automated Self-Assessment to Check PASCAL Programs for Syntax, Logic and Style Errors. Proceedings of the 26th SIGCSE Technical Symposium on Computer Science Education. ACM Press.
- [4] Helmick, M.T. 2007. Interface-Based Programming Assignments and Automatic Grading of Java Programs. Proceedings of ITiCSE '07. ACM Press.
- [5] Koike, H., Morita, H., Akama, K., Miura, K., 2006. Using an Automatic Marking System for Programming Course. Proceedings of SIGUCSS'06. ACM Press
- [6] McCabe, T.J. A Complexity Measure. 1976. IEEE Transaction on Software Engineering, SE-2(4), pp 308-320.
- [7] Halstead, M.H. 1977. Elements of Software Science. Elsevier Computer Science Library.
- [8] Zolnowski, J.M, and Simmons, D.B. 1977. A Complexity Measure Applied to FORTRAN. Proceedings of COMPSAC, pg 133-141.
- [9] Littlewood, B. 1978. How to Measure Software Reliability, and How Not To.. Proceeding of Third International Conference on Software Engineering. Pg 37-45.
- [10] DeMarco, I.J. 1977. Managing the Acquisition of Quality Computer Software. Proceedings of Sixth Annual Technical Symposium. ACM and National Bureau of Standards, Gaithersburg.
- [11] Bowen, J.B and Hugh-Fullerton. Are Current Approaches Sufficient for Measuring Software Quality?
- [12] Takahashi, R. 1997. Software Quality Classification Model Based on McCabe's Complexity Measure. Elsevier Science Inc.
- [13] Ravindran C. Basic and Criteria for Good Software Programming. [Http://Ezine.com](http://Ezine.com)

- [15] Norshuhani, Z., Emy Elyanee, M., Savita, K.S., Mazlina, M., Ellia, A. (2006) "Development of A Web-Based Automated Grading System for programming Assignments using Static Analysis Approach", International Conference on Technology and Operations Management, ICTOM 2006, Institut Teknologi Bandung, Indonesia, 1- 2 December 2006.
- [16] http://en.wikipedia.org/wiki/Dynamic_program_analysis
- [17] <http://www.umsl.edu/~sauterv/analysis/prototyping/proto.html>

APPENDIX A- Gantt Chart

Gantt Chart & Milestone

No	Activity	JULY				AUGUST				SEPTEMBER				OCTOBER				NOVEMBER
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	Algorithm Development																	
	User Interface Development																	
	Algorithm Testing																	
	Seminar																	
	Final Report Submission (First Draft)																	
	Pre-EDX																	
	Oral Presentation																	
	Submission of Dissertation																	

LEGEND

	Completed
	To be completed

Fig. 4.14: Progress Gantt Chart

Appendix B- Programming Assignments' Quality Measurement Criterion

Programming Assignments' Quality Measurement Criterion

I am Muhd Zulhafriz Aadel bin Marzuki (6788), currently doing my Final Year Project I for semester Jan08. This is a questionnaire on choosing the relevant criterion for assessing programming code in C language which objectives is to verify the right attributes used in manual assessment. The outcome expected from this questionnaire will help me to create an algorithm to automatically evaluate the quality of programming assignments by awarding marks to the particular codes. Hence, my project is hoped to significantly reduce marking workload for CIS lecturers who handling TAB1013 Structured Programming course in UTP.

Please rank the importance of these criterion individually in the boxes provided (1 for lowest, 5 for highest).

Criterion

Ranks

1 2 3 4 5

1. Input checking

--	--	--	--	--

Notes: This deals with the ability to intercept and stop program from running if invalid data types was entered by user. e.g. appropriate error handler.

1 2 3 4 5

2. Number of variables used

--	--	--	--	--

Notes: This deals with the number of variables used to derive to an answer (a student may use 4 variables (Fig. 1) while other may use 3 variables (Fig. 2) to obtain the same result). Fundamentally, fewer variables used the speed of the program increases.

```

int A, B, C, D;

A = 1;

B = 1;

C = A + B;

D = C; /* value of D is 2*/

```

Fig. 1

```

int A, B, D;

A = 1;

B = 1;

D = A + B; /* Value of D is 2*/

```

Fig. 2

3. Data types

1	2	3	4	5

Notes: This emphasized the usage of specific data type used. E.g. Float or Double data type instead of integer (e.g. the data type used should be able to accommodate the size of particular values (fig. 4)).

```

Long i, k;
Short j;

i = 15000000;
k = 25000000;

j = i + k;

/* memory for j is */
/* insufficient to */
/* store value i + k.*/

```

Fig. 3

```

Long i, k;
Short j;

j = 15;
k = 20;

i = k + j;

```

Fig. 4

Criterion

Ranks

1 2 3 4 5

--	--	--	--	--

4. Comments

Notes: Whether the codes built by students comes with a comments that describes their program briefly and its relevance (Fig. 5) to the program.

```
scanf("%d", &times.hour);    /* Enter hour here */  
printf("Enter hour: "); /* display Enter Hour: */
```

Fig. 5

```
scanf("%d", &times.hour);    /* This is a programming assignment */  
printf("Enter hour: "); /* You will get a correct answer */
```

Fig. 6: Unrelated comments

1 2 3 4 5

--	--	--	--	--

5. Program Neatness

Notes: Does the particular student starts the next line properly (Fig. 7) or lump all the codes into one line (Fig. 8)?

```
printf("Enter hour: ");  
scanf("%d", &times.hour);
```

fig. 7

```
printf("Enter hour: ");scanf("%d",&hour);
```

fig. 8

Criterion**Ranks**

6. Correctness- Output generated

1	2	3	4	5

Notes: This aspect deals with the ability to generate the expected output (Obtained the correct outcome logically (Fig.9)).

```
int a, b, c;  
printf("/nEnter two numbers : ");  
scanf("%d%d",&a,&b);  
c = a + b;  
printf("/nNow sum is now %d", &c);
```

Fig. 9

7. Correctness- Logic

1	2	3	4	5

Notes: Logic criteria deal with the structure and output of a loop. The loop structure can be executed, but it doesn't have upper limit causing it to loop endlessly (Fig. 10).

```
for (k=2; k>0; ++k)/*This loop would not stop looping*/  
{  
printf ("Inner %3d%3d/n", i,k);  
}
```

Fig. 10

Criterion**Ranks****1 2 3 4 5**

8. Correctness- Syntax Error

--	--	--	--	--

Notes: Syntax error is basically dealing with the error generated by students while doing their assignments. Students' codes were not complying with the C language's syntax (Fig. 11).

```
/* The 'print' command line should be 'printf'*/  
print("Inner %3d%3d/n", i,k);
```

Fig. 11

Notes:

Distribution date: 14th April 2008Collection Date: 21st April 2008

For all your of your cooperation, I would like to express my heartiest gratitude.
Hopefully, all the time spent on completing this question will aid on the completion of my project for the benefit of all. Thank you.

Appendix C- Algorithm for Measuring C Programming Assignments Quality Soft Codes

```
Imports System
Imports System.IO
Imports System.Text
Imports System.Threading
Imports System.Threading.Thread
Imports System.IO.Directory
Imports System.IO.File

Public Class Form1
    Private input As FileStream
    Private filereader As StreamReader
    Private output As FileStream
    Private filewriter As StreamWriter

    Dim input1 As String
    Dim input2 As String
    Dim countererror As Integer
    Dim flagStudAns As Integer
    Dim flagAnsSch As Integer
    Dim lineCount1 As Integer
    Dim lineCount2 As Integer
    Dim funcCompare As Integer

    Dim tempdataSplit() As String = {}
    Dim tempDataSplit2() As String = {}
    Dim linesfunc1() As String = {}
    Dim linesfunc2() As String = {}
    Dim tempFunc() As String = {}
    Dim commentArray() As String = {}
    Private Sub OpenFileDialog1_FileOk(ByVal sender As System.Object,
    ByVal e As System.ComponentModel.CancelEventArgs)

        End Sub
        Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

            btnGradNow.Enabled = False
            btnAnalysis.Enabled = False

        End Sub

        Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
            End Sub
            '-----Preprocessing Function-----
            '-----Subfunction - To set to lowercase all codes at
students' answer-----
            Sub ToLowerCase(ByVal sender As System.Object, ByVal e As
System.EventArgs)

                Dim filenameCreate = "StudAnsalter.txt"
                output = New FileStream(filenameCreate, FileMode.Create)
                Dim filewriter As New StreamWriter(output)
```

```

Dim i As Integer

Dim schoutput As System.IO.FileStream
Dim SchfileCreate = "SchAnsalter.txt"
schoutput = New FileStream(SchfileCreate, FileMode.Create)
Dim Schfilewriter As New StreamWriter(schoutput)
Dim x As Integer

For Each line As String In tempDataSplit2
    If tempDataSplit2(i).Length > 0 Then
        filewriter.WriteLine(tempDataSplit2(i).ToLower)
        Dim testdisp As String = tempDataSplit2(i).ToLower
    End If
    i += 1
Next
filewriter.Close()
output.Close()

For Each Schline As String In tempdataSplit
    If tempdataSplit(x).Length > 0 Then
        Schfilewriter.WriteLine(tempdataSplit(x).ToLower)
        Dim testdisp As String = tempdataSplit(x).ToLower
    End If
    x += 1
Next
Schfilewriter.Close()
schoutput.Close()

End Sub

'-----Subfunction - Replace all strings at students' answer--
-----
Sub ReplaceString(ByVal sender As System.Object, ByVal e As
System.EventArgs)
    Dim Studtempdatasplit() As String = {}
    Dim Schtempdatasplit() As String = {}
    Dim pos_1, pos_2 As Integer
    Dim placel, place2 As String
    Dim key As String
    Dim keyword As String
    Dim keywordcount() As String = {}
    Dim keywordcount2() As String = {}

    '-----Open Stud Answer File-----
    -----
    Dim filenameRead = "StudAnsalter.txt"
    input = New FileStream(filenameRead, FileMode.Open,
FileAccess.Read)
    filereader = New StreamReader(input)
    Dim tempData = filereader.ReadToEnd()
    Studtempdatasplit = Split(tempData, vbCrLf)
    filereader.Close()
    input.Close()

    Dim filenameCreate = "StudAnsalter.txt"
    output = New FileStream(filenameCreate, FileMode.Create)
    Dim filewriter As New StreamWriter(output)
    Dim i As Integer

```

```

'-----close student Answer file/ Open Answer
Scheme File-----'

Dim SchfilenameRead = "SchAnsalter.txt"
Dim schinput As System.IO.FileStream
Dim schfilereader As System.IO.StreamReader
schinput = New FileStream(SchfilenameRead, FileMode.Open,
FileAccess.Read)
schfilereader = New StreamReader(schinput)
Dim schtempData = schfilereader.ReadToEnd()
Schtempdatasplit = Split(schtempData, vbCrLf)

schfilereader.Close()
schinput.Close()

Dim schoutput As System.IO.FileStream
Dim SchfileCreate = "SchAnsalter.txt"
schoutput = New FileStream(SchfileCreate, FileMode.Create)
Dim Schfilewriter As New StreamWriter(schoutput)
Dim x As Integer

'-----Close Answer Scheme File-----
-----'
Dim foundKey As Boolean
Dim foundkey2 As Boolean

For Each line As String In Studtempdatasplit

    keywordcount2 = Split(Studtempdatasplit(i), "%")
    keyword = Studtempdatasplit(i).IndexOf("%")
    pos_1 = line.IndexOf(" ")
    place1 = line.Substring(0, pos_1 + 1)
    place2 = Mid(line, pos_1 + 2)
    pos_2 = place2.IndexOf(" ")

    If pos_2 > 0 Then
        key = place2.Substring(0, pos_2)
        keyword = Studtempdatasplit(i).Substring(0, keyword +
1)

        foundKey = True
    Else
        foundKey = False
    End If

    If foundKey = True Then
        filewriter.WriteLine(line.Replace(key, "freetext"))
    Else
        filewriter.WriteLine(line)
    End If
    i += 1
Next

For Each line As String In Schtempdatasplit

    keywordcount = Split(Schtempdatasplit(x), "%")
    keyword = Schtempdatasplit(x).IndexOf("%")

```

```

pos_1 = line.IndexOf(" ")
place1 = line.Substring(0, pos_1 + 1)
place2 = Mid(line, pos_1 + 2)
pos_2 = place2.IndexOf(" ")

If pos_2 > 0 Then
    key = place2.Substring(0, pos_2)
    keyword = Schtempdatasplit(x).Substring(0, keyword + 1)
    foundkey2 = True
Else
    foundkey2 = False
End If

If foundkey2 = True Then
    Schfilewriter.WriteLine(line.Replace(key, "freetext"))
Else
    Schfilewriter.WriteLine(line)
End If
x += 1
Next

filewriter.Close()
output.Close()

Schfilewriter.Close()
schoutput.Close()

End Sub
Sub DeleteComment(ByVal sender As System.Object, ByVal e As
System.EventArgs)
    '-----Open Stud Answer Start-----
    '-----'
    Dim Studtempdatasplit() As String = {}
    Dim schtempdatasplit() As String = {}

    Dim filenameRead = "StudAnsalter.txt"
    input = New FileStream(filenameRead, FileMode.Open,
FileAccess.Read)
    filereader = New StreamReader(input)
    Dim tempData = filereader.ReadToEnd()
    Studtempdatasplit = Split(tempData, vbCrLf)
    filereader.Close()
    input.Close()

    Dim filenameCreate = "StudAnsalter.txt"
    output = New FileStream(filenameCreate, FileMode.Create)
    Dim filewriter As New StreamWriter(output)
    Dim i As Integer
    '-----close student Answer file/ Open Answer
Scheme File-----'

    Dim SchfilenameRead = "SchAnsalter.txt"
    Dim schinput As System.IO.FileStream
    Dim schfilereader As System.IO.StreamReader
    schinput = New FileStream(SchfilenameRead, FileMode.Open,
FileAccess.Read)

```

```

schfilereader = New StreamReader(schinput)
Dim schtempData = schfilereader.ReadToEnd()
Schtempdatasplit = Split(schtempData, vbCrLf)

schfilereader.Close()
schinput.Close()

Dim schoutput As System.IO.FileStream
Dim SchfileCreate = "SchAnsalter.txt"
schoutput = New FileStream(SchfileCreate, FileMode.Create)
Dim Schfilewriter As New StreamWriter(schoutput)
Dim x As Integer

'-----close Answer Scheme File-----
-----

For Each lineComment As String In Studtempdatasplit
    Studtempdatasplit(i) = Trim(Studtempdatasplit(i))
    If Studtempdatasplit(i).StartsWith("/") And
Studtempdatasplit(i).EndsWith("/") Then

'filewriter.Write(Studtempdatasplit(i).Replace(Studtempdatasplit(i),
""))

        ElseIf Not Studtempdatasplit(i).StartsWith("/") And
Studtempdatasplit(i).EndsWith("/") And _
        (Studtempdatasplit(i).Contains("/") And
Studtempdatasplit(i).StartsWith("")) Then
            commentArray = Split(Studtempdatasplit(i), "/")
            filewriter.WriteLine(commentArray(0))
        ElseIf Studtempdatasplit(i).Contains("/") And Not
Studtempdatasplit(i).EndsWith("/") Then
            commentArray = Split(Studtempdatasplit(i), "/")
            filewriter.Write(commentArray(0))
            filewriter.Write("")
            filewriter.WriteLine(commentArray(2))
        Else
            filewriter.WriteLine(Studtempdatasplit(i))
        End If
        i += 1
Next

For Each lineComment As String In schtempdatasplit
    schtempdatasplit(x) = Trim(schtempdatasplit(x))
    If schtempdatasplit(x).StartsWith("/") And
schtempdatasplit(x).EndsWith("/") Then

        ElseIf Not schtempdatasplit(x).StartsWith("/") And
schtempdatasplit(x).EndsWith("/") And _
        (schtempdatasplit(x).Contains("/") And
schtempdatasplit(x).StartsWith("")) Then
            commentArray = Split(schtempdatasplit(x), "/")
            Schfilewriter.WriteLine(commentArray(0))
        ElseIf schtempdatasplit(x).Contains("/") And Not
schtempdatasplit(x).EndsWith("/") Then
            commentArray = Split(schtempdatasplit(x), "/")
            Schfilewriter.Write(commentArray(0))

```



```

        Schfilewriter.Write("")
        Schfilewriter.WriteLine(commentArray(2))
    Else
        Schfilewriter.WriteLine(schtempdatasplit(x))
    End If
    x += 1
Next
filewriter.Close()
output.Close()

Schfilewriter.Close()
schoutput.Close()

End Sub

'-----Subfunction - Delete the empty lines at students'
answer-----
Sub DeleteEmptyLine(ByVal sender As System.Object, ByVal e As
System.EventArgs)
    Dim i As Integer = 0
    Dim x As Integer = 0
    Dim Studtempdatasplit() As String = {}
    Dim schtempdatasplit() As String = {}

    '-----Open Stud Answer Start-----
    -----'
    Dim filenameRead = "StudAnsalter.txt"
    input = New FileStream(filenameRead, FileMode.Open,
FileAccess.Read)
    filereader = New StreamReader(input)
    Dim tempData = filereader.ReadToEnd()
    Studtempdatasplit = Split(tempData, vbCrLf)
    filereader.Close()
    input.Close()

    Dim filenameCreate = "StudAnsalter.txt"
    output = New FileStream(filenameCreate, FileMode.Create)
    Dim filewriter As New StreamWriter(output)

    '-----Open file student Ans End/ Answer Scheme
Start-----'
    Dim schinput As System.IO.FileStream
    Dim schfilereader As System.IO.StreamReader
    Dim SchfilenameRead = "SchAnsalter.txt"
    schinput = New FileStream(SchfilenameRead, FileMode.Open,
FileAccess.Read)
    schfilereader = New StreamReader(schinput)

    Dim schtempData = schfilereader.ReadToEnd()
    schtempdatasplit = Split(schtempData, vbCrLf)

    schfilereader.Close()
    schinput.Close()

    Dim schoutput As System.IO.FileStream
    Dim SchfileCreate = "SchAnsalter.txt"
    schoutput = New FileStream(SchfileCreate, FileMode.Create)
    Dim Schfilewriter As New StreamWriter(schoutput)

```

```

'-----Open Answer Scheme End-----
'

    For Each deleteLine As String In Studtempdatasplit
        If Studtempdatasplit(i).Length > 0 Or Studtempdatasplit(i)
<> "" Then
            filewriter.WriteLine(Studtempdatasplit(i))
        Else
        End If
        i += 1
    Next deleteLine

    For Each deleteLine As String In schtempdatasplit
        If schtempdatasplit(x).Length > 0 Or schtempdatasplit(x) <>
"" Then
            Schfilewriter.WriteLine(schtempdatasplit(x))
        Else
        End If
        x += 1
    Next deleteLine

    filewriter.Close()
    output.Close()

    Schfilewriter.Close()
    schoutput.Close()

End Sub
'-----Subfunction - Left justified all lines at students'
answer-----
Sub LeftJustified(ByVal sender As System.Object, ByVal e As
System.EventArgs)
    Dim Studtempdatasplit() As String = {}
    Dim schtempdatasplit() As String = {}
    Dim i As Integer
    Dim x As Integer

    '-----Open Stud Answer Start-----
    '
    Dim filenameRead = "StudAnsalter.txt"
    input = New FileStream(filenameRead, FileMode.Open,
FileAccess.Read)
    filereader = New StreamReader(input)
    Dim tempData = filereader.ReadToEnd()
    Studtempdatasplit = Split(tempData, vbCrLf)

    filereader.Close()
    input.Close()

    Dim filenameCreate = "StudAnsalter.txt"
    output = New FileStream(filenameCreate, FileMode.Create)
    Dim filewriter As New StreamWriter(output)

    '-----Open file student Ans End/ Answer Scheme
Start-----

```

```

Dim SchfilenameRead = "SchAnsalter.txt"
Dim schinput As System.IO.FileStream
Dim schfilereader As System.IO.StreamReader
schinput = New FileStream(SchfilenameRead, FileMode.Open,
FileAccess.Read)
schfilereader = New StreamReader(schinput)
Dim schtempData = schfilereader.ReadToEnd()
schtempdatasplit = Split(schtempData, vbCrLf)

schfilereader.Close()
schinput.Close()

Dim schoutput As System.IO.FileStream
Dim SchfileCreate = "SchAnsalter.txt"
schoutput = New FileStream(SchfileCreate, FileMode.Create)
Dim Schfilewriter As New StreamWriter(schoutput)

'-----Open Answer Scheme End-----
-----'

Dim foundBlank As Boolean
For Each line As String In Studtempdatasplit
    If Studtempdatasplit(i).Length > 0 Then
        filewriter.WriteLine(Studtempdatasplit(i).TrimStart)
        foundBlank = False
    Else
        If Not foundBlank Then
            foundBlank = True
        End If
    End If
    i += 1
Next

Dim foundBlank2 As Boolean
For Each line As String In schtempdatasplit
    If schtempdatasplit(x).Length > 0 Then
        Schfilewriter.WriteLine(schtempdatasplit(x).TrimStart)
        foundBlank2 = False
    Else
        If Not foundBlank2 Then
            foundBlank2 = True
        End If
    End If
    x += 1
Next

filewriter.Close()
output.Close()

Schfilewriter.Close()
schoutput.Close()

End Sub
'-----Subfunction - Replace all variables at students'
answer-----

```

```

    Sub ReplaceVariable(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Dim Studtempdatasplit() As String = {}
        Dim lines() As String = {}
        Dim posv_1, posv_2 As Integer
        Dim strvTemp As String
        Dim place1, place2, place3 As String
        Dim innerloop() As String = {}

        Dim filenameRead = "StudAnsalter.txt"
        input = New FileStream(filenameRead, FileMode.Open,
FileAccess.Read)
        filereader = New StreamReader(input)
        Dim tempData = filereader.ReadToEnd()
        Studtempdatasplit = Split(tempData, vbCrLf)
        filereader.Close()
        input.Close()

        Dim filenameCreate = "StudAnsalter.txt"
        output = New FileStream(filenameCreate, FileMode.Create)
        Dim filewriter As New StreamWriter(output)
        Dim i As Integer
        Dim j As Integer

        For Each line As String In Studtempdatasplit
            If Studtempdatasplit(i).Contains("(") And
Studtempdatasplit(i).Contains(")") And _
                Studtempdatasplit(i).Contains("scanf") Then
                innerloop = Split(Studtempdatasplit(i), ",")
                Dim arrUbound As Integer = UBound(innerloop)
                If innerloop(arrUbound).StartsWith("&") Then
                    Dim tempstore As Integer =
innerloop(arrUbound).IndexOf(")")
                    Dim storebal As String = Mid(innerloop(arrUbound),
tempstore + 1)

                        End If
                    End If
                Next

                filewriter.Close()
                output.Close()

        End Sub

'-----compare the amount of variables declare by students
answer-----'
    Sub VarQuantityCheck(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Dim lines1() As String = {}
        Dim lines2() As String = {}
        'Dim i As Integer
        Dim y As Integer
        Dim flag As Integer = 0
        Dim varNum As Integer
        Dim varNumsc As Integer
        Dim FileName1 As String = input1
        Dim FileName2 As String = input2

```

```

        For Each filetext1 As String In tempDataSplit2
            If tempDataSplit2(y).Contains("dim") Or
tempDataSplit2(y).Contains("int") Or _
                tempDataSplit2(y).Contains("float") Or
tempDataSplit2(y).Contains("short") Or _
                tempDataSplit2(y).Contains("long") Or
tempDataSplit2(y).Contains("char") Or _
                tempDataSplit2(y).Contains("double") And Not
(tempDataSplit2(y).Contains("(") And _
tempDataSplit2(y).Contains(")")) Then
                varNum += 1
            End If
            y += 1
        Next filetext1
        y = 0
        For Each line2 As String In tempdataSplit
            If tempdataSplit(y).Contains("dim") Or
tempdataSplit(y).Contains("int") Or _
                tempdataSplit(y).Contains("float") Or
tempdataSplit(y).Contains("short") Or _
                tempdataSplit(y).Contains("long") Or
tempdataSplit(y).Contains("char") Or _
                tempdataSplit(y).Contains("double") And Not
(tempdataSplit(y).Contains("(") And _
tempdataSplit(y).Contains(")")) Then
                varNumsc += 1
            End If
            y += 1
        Next line2

End Sub

'-----Subfunction - Do the comparison process-----
Sub modComparisonApproach(ByVal sender, ByVal e)

    counterror = 0

    Dim x As Integer = 0
    Dim y As Integer = 0

    counterror = 0
    For Each line2 As String In tempdataSplit
        For Each line1 As String In tempDataSplit2

            If tempDataSplit2(x).ToString =
tempdataSplit(y).ToString Then
                Else
                    MessageBox.Show(tempDataSplit2(x).ToString)
                End If
                x += 1
            Next line1
            y += 1
        Next line2

    End Sub

```

```

Sub functionSearch(ByVal sender As System.Object, ByVal e As
System.EventArgs)
    Dim x As Integer = 0
    Dim y As Integer = 0
    Dim funcCount() As String = {}

    countererror = 0
    For Each line1 As String In tempDataSplit2

        If (tempDataSplit2(x).StartsWith("int") Or
tempDataSplit2(x).StartsWith("char") Or _
tempDataSplit2(x).StartsWith("float") Or
tempDataSplit2(x).StartsWith("double") Or _
tempDataSplit2(x).StartsWith("long") Or
tempDataSplit2(x).StartsWith("short") Or _
tempDataSplit2(x).StartsWith("void")) And
(tempDataSplit2(x).Contains("(") And tempDataSplit2(x).Contains(")"))
Then
            Dim tempfunc = Split(tempDataSplit2(x), " ")
            Dim elementCount As String = funcCount.Length
            If elementCount = 0 Then
                funcCount(0) = tempfunc(1)
            Else
                funcCompare = StrComp(funcCount(x - 1),
tempfunc(1))
                If funcCompare <> 0 Then
                    funcCount(x) = tempfunc(1)
                End If
            End If
            End If
            x += 1
        Next line1
        For Each funcloop As String In funcCount

            funcloop = funcloop + 1
        Next funcloop

    End Sub

Private Sub syntaxCheck(ByVal sender As System.Object, ByVal e As
System.EventArgs)
    Dim lines1() As String = {}

    Dim FileName1 As String = input1

    If File.Exists(FileName1) Then

        Dim srl As New StreamReader(FileName1)
        Dim fileText1 As String = srl.ReadToEnd()
        srl.Close()
        lines1 = Split(fileText1, vbCrLf)

        Dim fs1 As New FileStream(input1, FileMode.Create)
        Dim sw1 As New StreamWriter(fs1)

        Dim x As Integer = 0
        Dim y As Integer = 0

```

```

        If Not lines1(0).StartsWith("#include <stdio.h>") Then
            countError += 1
        End If

        x = 2
        For Each line1 As String In lines1
            If lines1(x).Contains("main") Then
                If Not lines1(x).StartsWith("int") Then
                    countError += 1
                End If
                If Not lines1(x).EndsWith("(") Then
                    countError += 1
                End If
            End If
            x += 1
        Next line1
    End If

End Sub

Private Sub btnMark_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSelAns.Click
    Dim Studtempdatasplit() As String = {}
    Dim schtempdatasplit() As String = {}

    MessageBox.Show("Please Select a Student's Answer")
    Dim filechooser As New OpenFileDialog()
    Dim result2 As DialogResult = filechooser.ShowDialog()
    Dim filenameStudAns As String

    MessageBox.Show("Please Select an Answer Scheme")
    Dim filechooserAnsSch As New OpenFileDialog()
    Dim result As DialogResult = filechooserAnsSch.ShowDialog()
    Dim filenameAnsSch As String

    '-----Upload Students Answer-----'
    If result2 = Windows.Forms.DialogResult.Cancel Then
        Return
    End If

    filenameStudAns = filechooser.FileName

    If filenameStudAns = "" Or filenameStudAns Is Nothing Then
        MessageBox.Show("Invalid file name", "Error", _
        MessageBoxButtons.OK, MessageBoxIcon.Error)
        flagStudAns = 0
    Else
        input = New FileStream(filenameStudAns, FileMode.Open,
FileAccess.Read)
        filereader = New StreamReader(input)
        Dim tempData2 = filereader.ReadToEnd()
        txtStudAns.Text = tempData2
        tempDataSplit2 = Split(tempData2, vbCrLf)
        filereader.Close()
        input.Close()
    End If

```

```

        End If
        '-----Upload Students Answer
End-----'

        '-----Upload Answer Scheme
File-----'
        If result = Windows.Forms.DialogResult.Cancel Then
            Return
        End If

        filenameAnsSch = filechooserAnsSch.FileName

        If filenameAnsSch = "" Or filenameAnsSch Is Nothing Then
            MessageBox.Show("Invalid file name", "Error", _
                MessageBoxButtons.OK, MessageBoxIcon.Error)
            flagAnsSch = 0

        Else
            input = New FileStream(filenameAnsSch, FileMode.Open,
FileAccess.Read)
            filereader = New StreamReader(input)
            Dim tempData = filereader.ReadToEnd()
            txtAnsSch.Text = tempData
            tempdataSplit = Split(tempData, vbCrLf)
            flagAnsSch = 1
            filereader.Close()

        End If
        '-----Upload Answer Scheme File end-----
-----'

        ToLowerCase(sender, e)
        ReplaceString(sender, e)
        DeleteComment(sender, e)
        DeleteEmptyLine(sender, e)
        LeftJustified(sender, e)
        ReplaceVariable(sender, e)

        Dim studinput As System.IO.FileStream
        Dim studfilereader As System.IO.StreamReader
        Dim studfilenameRead = "StudAnsalter.txt"
        studinput = New FileStream(studfilenameRead, FileMode.Open,
FileAccess.Read)
        studfilereader = New StreamReader(studinput)
        txtStudAnsAlt.Text = studfilereader.ReadToEnd()
        studinput.Close()
        studfilereader.Close()

        Dim SchfilenameRead = "SchAnsalter.txt"
        Dim schinput As System.IO.FileStream
        Dim schfilereader As System.IO.StreamReader
        schinput = New FileStream(SchfilenameRead, FileMode.Open,
FileAccess.Read)
        schfilereader = New StreamReader(schinput)
        txtAnsSchAlt.Text = schfilereader.ReadToEnd()

```



```

schinput.Close()
schfilereader.Close()

btnGradNow.Enabled = True

End Sub

' -----SubFunction- To evaluate the codes' neatness-----
Sub CodesNeatness(ByVal sender As System.Object, ByVal e As
System.EventArgs)
    Dim lines1() As String = {}
    Dim flag As Integer = 0
    Dim FileName1 As String = input1
    Dim filenameNeat As String = input2
    Dim i As Integer

    Dim sr2 As New StreamReader(filenameNeat)
    Dim sr1 As New StreamReader(FileName1)

    Dim wholefile = sr2.ReadToEnd
    Dim fileLine As Integer = wholefile.count

    For i = 0 To i = fileLine
        If File.Exists(FileName1) Then

            Dim fileText1 = sr1.ReadLine
            '-----check for loop statement neatness-----
            -----'

            If fileText1.contains(" for ") Then
                If fileText1.startswith("for ") Then
                    flag = flag + 1
                Else
                    flag = 0
                End If
            End If

            If fileText1.contains("next") Then
                If fileText1.startswith("next ") And
fileText1.endswith("next") Then '<--- advice
                    flag = flag + 1
                Else
                    flag = flag
                End If
            End If

            '-----For loop statement check ends-----
            -----'

            '-----check the do while loop statement's neatness---
            -----'

            If fileText1.contains(" do ") Then
                If fileText1.startswith("do ") Then
                    flag = flag + 1
                Else
                    flag = flag
                End If
            End If
        End If
    Next i
End Sub

```

```

        End If

        If fileText1.contains(" loop ") Then
            If fileText1.startswith("loop ") And
fileText1.endswith("loop") Then
                flag = flag + 1
            Else
                flag = flag - 1
            End If
        Else
            flag = flag
        End If
    '-----do while loop ends-----
--'

    If fileText1.contains(";") Then
        If fileText1.endswith(";") Then
            flag = flag + 1
        ElseIf Not fileText1.endswith(";") Then
            flag = flag
        End If
    End If

    srl.Close()

    End If
Next

End Sub
Private Sub btnAnsScheme_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs)
    End Sub

    Private Sub btnSave_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnGradNow.Click

        Dim Studtempdatasplit() As String = {}
        Dim schtempdatasplit() As String = {}
        Dim i As Integer
        Dim x As Integer

        '-----Open Stud Answer Start-----
        -----'
        Dim filenameRead = "StudAnsalter.txt"
        input = New FileStream(filenameRead, FileMode.Open,
FileAccess.Read)
        filereader = New StreamReader(input)
        Dim tempData = filereader.ReadToEnd()
        Studtempdatasplit = Split(tempData, vbCrLf)

        filereader.Close()
        input.Close()

        Dim filenameCreate = "StudAnsalter.txt"
        output = New FileStream(filenameCreate, FileMode.Create)
        Dim filewriter As New StreamWriter(output)

```

```

'-----Open file student Ans End/ Answer Scheme
Start-----'
Dim SchfilenameRead = "SchAnsalter.txt"
Dim schinput As System.IO.FileStream
Dim schfilereader As System.IO.StreamReader
schinput = New FileStream(SchfilenameRead, FileMode.Open,
FileAccess.Read)
schfilereader = New StreamReader(schinput)
Dim schtempData = schfilereader.ReadToEnd()
schtempdatasplit = Split(schtempData, vbCrLf)

schfilereader.Close()
schinput.Close()

Dim schoutput As System.IO.FileStream
Dim SchfileCreate = "SchAnsalter.txt"
schoutput = New FileStream(SchfileCreate, FileMode.Create)
Dim Schfilewriter As New StreamWriter(schoutput)

'-----Open Answer Scheme End-----
-----'

VarQuantityCheck(sender, e)
functionSearch(sender, e)
modComparisonApproach(sender, e)
syntaxCheck(sender, e)

btnAnalysis.Enabled = True

End Sub

Private Sub TextBox1_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TextBox1.TextChanged

End Sub
End Class

```