Creative Technology Department
Student Feedback – AY1920

Unit:	**Introduction To Programming**	**Name :**	**Sam Robinson**
Level :	**4**	**Mark:**	**97**
Assignment No:	**2**	**Marker :**	**Andrew Watson**
Hand in date:	**Friday January 24th, 2020**	**Q A :**	**Karsten Pedersen**
Dyslexia Marking Guidelines to be used :			

Lecturer Feedback:

GENERAL FEEDBACK COMMENT:

Congratulations for completing the first challenging programming assignment. This assignment was a designed to allow you to directly apply the knowledge that you have gained through your active participation with the lectures, the "BrightSpace" reading reference materials and with the labs particularly by completing the labs exercises. The assignment provided you with the opportunity to practice the analysis of a non-trivial problem and identify the main problem components using analysis diagrams, further you will have also used the analysis to understand how the identified components interact to create a functional solution. The components discovered in your analysis were further defined and detailed in the design section where you were able to begin to consider the data and operations used in the game logic and algorithms. The design section provided you with the opportunity to think about how the game logic and various algorithms could be designed and specified using further refined design techniques. The implementation section provided you with the opportunity to produce a functional solution of your design coded in C++. This section allowed you to show both your solution and your ability to translate your solution into code.

Analysis: The Analysis Section is to do with the analysis of the problem and its identification and decomposition into distinct modules/classes. The analysis should specify each module's function including inputs and outputs and their interaction with the system. The analysis section is to address the question of "What will the program do?" and will give an insight into the program's architecture/structure particularly the logic of the program and how it will function. The focus of the analysis is identifying modules and program structure / operation (not how they are coded). The analysis should use diagrams to illustrate the system. It appears that in general the analysis has been done after the program has been implemented. This is not the correct way to develop software. Many students have not provided enough information for both the analysis and design sections. Many of the analysis diagrams seemed to be bespoke, created by the report author, rather than using the standardised flowchart and structure diagram symbols. Many students discussed detailed implementation issues which are not appropriate at this point. Some of the flowcharts also have sections of isolated flows which do not tie in with main the program. Typically the analysis sections were short and didn't address the issues at the detail required. Also the flowcharts that were reverse engineered from the code tended to be incorrect However, this year did see an increase in number students that produced excellent analysis

Design: The Design section is to do with the detailed design of each of the modules/classes that have been identified in the analysis section. Each module's design will be expressed in terms of input, process and output with respect to the required functionality. The modules' functionality will be presented using flow charts , supporting design description and / or pseudo code. The design section should also discuss the "nuts and bolts" of how the game operates. It appears that in many instances that the design has been done after the program has been implemented. This is not the correct way to develop software. The design sections submitted were on the whole too short and did not contain sufficient level of design detail. The flowchart and pseudo code suffered from lack of organisation and use of the standardised symbols. Many students used the same diagrams for both the analysis and design.

Implementation: The Implementation section is to do with all aspects of the application's implementation. This section will contain commented source code for the application. The application's source code will be reviewed with respect to coding standards (layout formatting, comments), its structure and its logical separation of the program into modules .The executable will be assessed against the defined marking criteria particularly the user operation e.g. Controlled Exit, User Instructions and Usability. Checking user input for errors. The implementation

section will contain the testing procedures used during the testing of the application This section deals with the testing of the developed application implementation for its operation and compliance with the required specification. In general the majority of programs were implemented in a clear logical, structured and modular way. Several Students games did not contain required elements such as instructions or a controlled quit. Most programs met the majority of the specifications. Many programs were implemented using a simple menu based interface, which in general worked correctly. The main issues were to do with strict adherence to good coding standards and source layout. The comments tended to be very limited. Several programs exhibited problems created by the use of global variables and recursive function calls. Many programs did not include input data validation for type and range. There were several instances where C++ String and associated methods were used despite their used being disallowed.

SUMMARY FEEDBACK COMMENT:

Section	Feed Back	Mark
Self Assessment(5%)	Very good comments and reflections	5
Analysis(15%)	Sam this section is excellent well done	15
Design(15%)	Sam this section also excellent well done.	15
Test and Conclusions(5%)	This section is really good, you have reflected on the process, the development and problems you encountered.	5
Implementation(60%)	Sam, this a really excellent effort well done. All features and more. The code is well written, it is very modular and easy for me follow. The code is to very good standard The project is made o multiple source files. The game play is good and the gui is clear. The game meets and exceeds the assignment requirements. The code is well structured. There are opportunities to refactor7 repeated or common code. You have excellent input validation !!! Only comment is that you don't show the player the word they should have guessed when they lose. This was pleasure mark .Thanksyou well Done.	57
	Total	97

3 Feed Forward Suggestions:

(i) Keep what you are doing !!!

(ii) Review and refactor repeated or common code

(iii) Tell the user what the word was when get it wrong.

SIGNED (Marker): A.E.WATSON Date: 07/02/2020

MARKINGSCHEME:

Deliverable	0% -> 39%	40% -> 49%	50% -> 59%	60% -> 69%	70%+
Report 40%					
Self Assessment	• Not completed. •	• Partially completed	• Mostly completed.	• Mostly completed, some missing components.	• **Completed**
Analysis.	• No evidence Problem analysis • No evidence of Analysis Diagrams • No evidence of Clear Program Operation •	• Little evidence Problem analysis • Little evidence of Analysis Diagrams • Little evidence of Clear Program Operation •	• Evidence of Problem analysis • Evidence of Analysis Diagrams • Evidence of Clear Program Operation •	• Good evidence of Problem analysis • Good evidence of Analysis Diagrams • Good evidence of Clear Program Operation	• **Excellent Problem analysis** • **Excellent Analysis Diagrams** • **Excellent description of Clear Program Operation** •
Design	• No evidence of module design • No evidence of Design Techniques • No evidence of detailed design diagrams	• Little evidence of module design • Little evidence of Design Techniques • Little evidence of detailed design diagrams	• Evidence of module design • Evidence of Design Techniques • Evidence of detailed design diagrams	• Good evidence of module design • Good evidence of Design Techniques • Good evidence of detailed design diagrams	• **Excellent module design** • **Excellent of Design Techniques** • **Excellent evidence of detailed design diagrams**
Testing and Conclusions	• No test strategy applied. • No conclusions	• Limited test strategy applied • Short limited Conclusions	• Basic test strategy applied • Basic conclusions	• Good test strategy applied • Good Conclusions	• **Excellent test strategy applied** • **Excellent Conclusions**
Implementation 60%					
Features and Functionality; Adherence to coding standards; Code Quality; Design and structure; Readability and Comments	• Global variables used • No evidence of modularity. • No evidence of programming style. • No test strategy applied. • No understanding of program construction. • No application of BU Coding Standards. • No real evidence of meeting system requirements (i.e. Features and Functionality as described in the brief)	• Functions with no parameters • Little evidence of modularity • Little evidence of programming style • Limited test strategy applied • Limited understanding of program construction • Limited application of BU coding standards • Some evidence of meeting system requirements (i.e. Features and Functionality as described in the brief)	• Functions with limited parameters • Some evidence of modularity • Some evidence of the use of programming style • Basic test strategy applied • Some understanding of program construction • Reasonable application of BU Coding Standards • Reasonable evidence of meeting system requirements (i.e. Features and Functionality as described in the brief)	• Good use of function parameters • Good evidence of modularity • Good evidence of the use of programming style • Good test strategy applied • Good understanding of program construction • **Good application of BU Coding standards** • Good evidence of meeting system requirements (i.e. Features and Functionality as described in the brief)	• **Excellent use of function parameters** • **Excellent modularity** • **Excellent programming practices** • **Excellent test strategy applied** • **Comprehensive understanding of program construction** • Full application of BU Coding Standards • Fully demonstrated all system **requirements (i.e. Features and Functionality as described in the brief)**

Element	Contribution.	100% Mark	Weighted Mark
Self-Assessment	**5%**	**100**	**5**
Analysis	**15%**	**100**	**15**
Design	**15%**	**100**	**15**
Testing and Conclusions	**5%**	**100**	**5**
Implementation	**60%**	**95**	**57**