Faculty of Science and Technology

Department of Creative Technology

Academic Year 2019/2020

Unit Name: Introduction To Programming		
Assignment Title: Code and Report		
Assignment Number: 2 of 2		
Course/s: BSc Games Software Engineering	**Unit Level:** 4	**Unit Credit Value:** 20
Primary Marker: Andrew Watson	**Quality Assessor:** Karsten Pedersen	
Assignment Issue Date: Monday October 14th, 2019	**Assignment Submission Date:** Friday January 24th, 2020 **Time:** 12:30pm	
Individual or Group Information: This is an **Individual** assignment which carries **70%** of the final unit mark		
Feedback Method: Via Brightspace	**Assignment Weighting:** 70% of the final unit mark	

SUBMISSION METHOD(S)

1) **Electronic Submission**

Assignment is to be electronically submitted by 12:30pm on the due date (please allow sufficient time to upload files before the deadline) via:

(i) Large File Submission Link on Brightspace

THE ASSESSMENT TASK
This is an individual assignment.

To demonstrate your understanding of both program design concepts and the C++ programming language you are required to design and implement a console-based “Hangman” Game application, programmed in C++.

You must also write a technical report to document the analysis, design and implementation of the software.

The assignment will be assessed on the Analysis, Design and the Implementation of the solution.

For this assignment, you will write a program to play the game “Hangman”.

The application is required to allow one player to play “Hangman”.

The objective of this game is to prevent the Man from being hanged. This is achieved by the player correctly guessing an unknown word or phrase as specified by the Executioner.

The computer will act as executioner.

The designer should develop an appropriate interface for the game. The interface should include the following information:

- The Gallows: Displaying the current state of the execution
- The Word /Phrase: Display the correctly guessed letters in their correct positions and also indicate the remaining unknown letters positions
- Number guesses / lives left
- How to Play / Game Rules
- The number of player’s guesses remaining
- A game exit

Other games features that the designer may consider including are:

- A scoring system
- An output of the all the letters already guessed
- A level option where single words and phrases are on different levels of complexity
- Theme based Hangman
- Cryptic Clue Hangman

After finishing a game, your program should ask the user if they want to play again or quit the program. The program should play another game unless the user chose to quit the program.

The designer should develop an appropriate game user interface for the game which should be clear and intuitive for the user.

Your program should provide a debug option that displays the secret word/phrase at the beginning of each round for grading/debugging purposes.

The interface should **clearly keep the user informed at all stages** of “what is required” and “what the game state is” (Do not assume the player knows what to do, you need to **prompt them**).

The application should be written in C++, using Microsoft Visual Studio VS 2017.

The application must be console based
The application must be menu based
The application must have a controlled start
The game must have instructions
The game must have a controlled exit

After playing a game the player should be allowed to select whether to play another game

Limitations: The man hangs after six incorrect guesses. (Standard Game).

Note: The software is only to be **STRUCTURED (procedural based)**, not Object based (i.e., NO classes or STL such as std::vector or std::string).

All strings must be CStrings (i.e. null terminated char arrays) and NOT the C++ String Class String.

Program Details:

You should use Microsoft Visual Studio 2017 C++ to meet the above requirements. The executable will be measured against the defined marking criteria (see Marking Scheme section below).

You will need to hand in all source code, project file. You will also need to make and submit a video demonstrating your game, which you can use to showcase any special features. This video must be in mp4 format. Please ensure the video is of a reasonable file size (by using compression).

The **Implementation section** is to do with all aspects of the application's implementation. The application's source code will be reviewed with respect to coding standards (layout formatting, comments), its design and structure and its logical separation of the program into modules and the use of parameter passing. The executable will be assessed against the defined marking criteria.

Report Details:

The student must complete the **Introduction to Programming assignment report template** in Appendix A .The document contains three main sections that should be used to discuss each aspect of the game's development stages: analysis, design and testing.

The software assignment report template contains three mains sections plus a self-assessment and reference section, which must be completed. You should thus be looking at the following:

- Self-Assessment
- Analysis
- Design
- Testing and Conclusions
- References

Each section of the report must be completed.

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.

The **Design section** is to do with the detailed design of each of the modules/classes that have been identified in the analysis section.

For this assignment you are only required to provide the detailed design for the module dealing with **testing** the inputted letter / word with the hidden word. The module's design will be expressed in terms of input, process and output with respect to the required functionality. The module 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 character testing operates.

Note: The module you design for testing the inputted character, may use other modules to achieve the appropriate testing, if this is the case these modules must also be detailed.

The **Testing and conclusion section** will contain an overview of the testing procedures used during the testing of the application. You should include some reflection on the development process and how the game performed relative to your initial ideas. It would also be useful to comment on any aspects that changed or were modified during the development. You should also include details of any problems that you encounter during the development and what you did to resolve them.

THE DELIVERABLE AND SUBMISSION FORMAT

You must submit the following items for assessment:

You must submit the following items for assessment:

Submission will be through the Online Submission System on Brightspace. You must submit a zipped folder containing:

- Game executable and assets (e.g. images, sounds)
- Microsoft Visual Studio project
- All source code .cpp and .h files
- A video recording of the game being played (in mp4 format, using compression)
- Completed Introduction to Programming report template document (see Appendix B)

THE SUBMISSION DEADLINE

You must submit your work by the following deadline:

Friday January 24th, 2020

The deadline for all submissions is 12:30pm unless otherwise stated.

THE MARKING CRITERIA

Your assignment will be assessed using the following marking criteria:

The Marking Scheme:

Your assignment will be assessed using the following mark scheme criteria:

Report		**40%**
	Self-assessment	5%
	Analysis	15%
	Design	15%
	Testing and Conclusions	5%
Implementation		**60%**
	Program Features and Functionality; Adherence to coding standards; Code Quality; Design and Structure; Readability and Comments	

See below for detailed marking scheme:

MARKINGSCHEME:

Deliverable	0% -> 39%	40% -> 49%	50% -> 59%	60% -> 69%	70%+
Report					
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					
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.**
Self-Assessment	**5%**
Analysis	**15%**
Design	**15%**
Testing and Conclusions	**5%**
Implementation	**60%**

THE LEARNING OUTCOMES:

This assignment will assess the following ILOs:

1. Review a simple software system using an appropriate design methodology;
2. Translate a software system from a written specification in a comprehensive and unambiguous manner;
3. Express skills in the planning and executing of the structural and functional testing and debugging of a software application;

Give examples of the use of computer algorithms, data structures and formats in software programs

QUESTIONS ABOUT THE ASSIGNMENT BRIEF:

You should address any questions about this assignment brief to the Unit Leader, whose details are shown below:

Unit Leader: **Andrew Watson**

Unit Leader Email: **awatson@bournemouth.ac.uk**

You can also ask questions in lectures and lab sessions.

ASSIGNMENT GUIDANCE NOTES – Academic Year 2019-2020

You must keep a copy of your assignment – the university will not take responsibility for lost assignments. Please make sure you back up your work carefully.

Submission Deadlines:

If a piece of coursework is not submitted by the required deadline, the following will apply:

1. If coursework is submitted within 72 hours after the deadline, the maximum mark that can be awarded is 40%. If the assessment achieves a pass mark and subject to the overall performance of the unit and the student's profile for the level, it will be accepted by the Assessment Board as the reassessment piece. The unit will count towards the reassessment allowance for the level; This ruling will apply to written coursework and artefacts only; This ruling will apply to the first attempt only (including any subsequent attempt taken as a first attempt due to exceptional circumstances).

2. If a first attempt coursework is submitted more than 72 hours after the deadline, a mark of zero (0%) will be awarded.

3. Failure to submit/complete any other types of coursework (which includes resubmission coursework without exceptional circumstances) by the required deadline will result in a mark of zero (0%) being awarded.

The Standard Assessment Regulations can be found on **Brightspace**.

Exceptional Circumstances:

If you have any valid **exceptional circumstances** which mean that you cannot meet an assignment submission deadline and you wish to request an extension, you will need to complete and submit the Exceptional Circumstances Form for consideration to your Programme Support Officer (based in C114) together with appropriate supporting evidence (e.g, GP note) normally **before the coursework deadline**. Further details on the procedure and the exceptional circumstances form can be found on **Brightspace**. Please make sure that you read these documents carefully before submitting anything for consideration. For further guidance on exceptional circumstances please see your Programme Leader.

Avoiding Plagiarism:

You must acknowledge your source every time you refer to others' work, using the **BU Harvard Referencing** system (Author Date Method). Failure to do so amounts to plagiarism which is against University regulations. Please refer to http://libguides.bournemouth.ac.uk/bu-referencing-harvard-style for the University's guide to citation in the Harvard style. Also be aware of Self-plagiarism, this primarily occurs when a student submits a piece of work to fulfill the assessment requirement for a particular unit and all or part of the content has been previously submitted by that student for formal assessment on the same/a different unit. Further information on academic offences can be found on **Brightspace** and from https://www1.bournemouth.ac.uk/discover/library/using-library/how-guides/how-avoid-academic-offences

Accessing Learning Support:

Students with **Additional Learning Needs** may contact Learning Support on www.bournemouth.ac.uk/als

ALL CODE WILL BE CHECKED FOR PLAGIARISM BOTH FROM EACH OTHER AND THE INTERNET

Ethical Compliance:

You should not be conducting any primary research (i.e. carrying out an investigation to acquire data first-hand, for example, where it involves approaching participants to ask questions or to participate in surveys, questionnaires, interviews, observations, focus groups, etc.) unless otherwise specified in the brief. However, if there is a genuine requirement to collect primary research data you will require ethical approval before doing so. In the first instance, please discuss with the Unit Leader. The collection of primary data without appropriate ethical approval is a serious breach of Bournemouth University's Research Ethics Code of Practice and will be treated as Research Misconduct.

Disclaimer:

The information provided in this assignment brief is correct at time of publication. In the unlikely event that any changes are deemed necessary, they will be communicated clearly via e-mail and Brightspace and a new version of this assignment brief will be circulated.

Assignment Reference: ITP_Ass2_AY1920_200919_FINAL_RELEASE

Appendix A

Bournemouth University

Faculty of Science and Technology

2019/2020

Level 4

Introduction To Programming

Assignment 2

HangMan Program

Analysis, Design, and Implementation

Report Template

1. Self-Assessment of Performance 5%

Tutor : Andrew Watson

Student ID	

Circle the appropriate response:

Did I submit the assignment on time?	**Yes**	**No**		
Did I complete the assignment?	**Yes**	**No**		
If not, approx. how much did I complete?	**%**			
How happy am I with what I submitted?	**Very happy**	**Satisfied**	**Disappointed**	**Ashamed**
What mark do I expect?	**%**			
Did I spend enough time on the assignment?	**Yes**	**No**		
Did I get it proof-read by someone else?	**Yes**	**No**		
Have I properly ‘referenced’ it?	**Yes**	**No**		
Could I improve the presentation?	**Yes**	**No**		

Answer the following questions:

The best part of my performance was:	
The worst part of my performance was:	
One way in which I could improve the content of my assignment is:	
One way in which I could improve the presentation of my assignment is:	
One thing I will do to improve my performance in my next assignment is:	
Another thing I will do to improve my performance in my next assignment is:	

2. Analysis : Marks 15%

Complete this section using Arial font 12pts

This section is to do with the analysis of the problem.
Break down the problem into distinct modules, indicating the inputs, processes and outputs.
This about what the program is required to do?
What is the logic of the program and how it will function?
Focus on identifying modules and program structure / operation, (not how they are coded). You should also think about the module interactions, is information passed between them.
Use diagrams to illustrate the analysis. Appropriate Diagrams
Diagrams: Top Level IPO diagram, Top level Structure Diagram ,Top level Flowchart ,Top Level Pseudocode Diagram

Guidance relating to Program Design and Diagramming is available on the VLE
***** This section in blue should be removed from the final report *****

3. Design : Marks 15%

Complete this section using Arial font 12pts

The design section is where each of the modules/classes that have been identified in the analysis section are then looked at in greater detail with respect to design of the required functionality.

The modules' design in terms of interfaces such as input and output and their functionality.

Use of Flow Charts and /or Detailed Pseudocode.

Guidance relating to Program Design ,Flow Charts ,Pseudo code is available on the VLE

***** This section in blue should be removed from the final report *****

4. Testing and Conclusions   (Marks: 5%)

Complete this section using Arial font 12pts

This section will contain an overview of the testing procedures used during your testing of the application. You should include some reflection on the development process and how the game performed relative to your initial ideas. It would also be useful to comment on any aspects that changed or were modified during the development. You should also include details of any problems that you encountered during the development and what you did to resolve them

This section in blue should be removed from the final report

Appendices .References

Complete this section using Arial font 12pts

This section should be used for all referenced materials. This can be Web sites, Tutorials, Books, Articles,

Source code, Acknowledgements and others.