CODELAB I

ASSESSMENT 2: Utility App

Module Coordinator: Jake Hobbs

Marker: Jake Hobbs

Contribution towards overall module mark	60%
Date set	November 25, 2018
Marked work returned by	February 14, 2019
DEADLINE DATE	January 24, 2019 23:59

Assessment 2: Utility App

The Brief

Your task is to create a Vending Machine program using the C++ programming language. The program should demonstrate your knowledge of programming and make use of the techniques introduced over the course of the module

Deliverables

The deliverables for this assignment are as follows:

- The Application: The C++ code (e.g. main.cpp file) and any supplementary assets required to run your vending machine. This should be submitted to your Github repository.
- The Development Document (1000+ words).

The Application

The Vending Machine must have the following features as a minimum requirement:

- A menu of drinks and snacks presented via the console. The number and range of items is up to you.
- A set of codes that the user can input to select a particular drink or snack.
- A way of capturing the user's inputted code.
- A way of managing money. The user should be able to input any amount of money and have the correct change returned.
- A message that tells the user that a particular drink or snack has been dispensed.
- A message that tells the user how much change they have received.
- Comments in the code to explain key operations.

You may wish to add additional features to your Vending Machine to achieve higher marks. Below is an indication of some of these features, however you may also wish to come up with your own:

- A method of categorising items in the vending machine to improve the user experience (e.g. 'Chocolate' or 'Hot Drinks').
- A way of allowing users to buy additional items if they have enough credit.
- An intelligence system for suggesting purchases. For example, if you buy a coffee, the vending machine may suggest that you buy biscuits.
- The use of functions to improve the structure of your program.
- A Graphical User Interface (GUI) that replaces console messages. This will require some self exploration as GUI implementation is not covered until CodeLab II. It is recommended that you use openFrameworks if you wish to deploy a GUI as this is used extensively in CodeLab II

The Development Document

Your Vending Machine C++ program must be accompanied by a Development Document of a minimum of 1000 words (there is no maximum word count).

Please include the following elements (note the suggested word counts):

- *Brief:* A short explanation of what you have been asked to build. This section should also include a link to your GitHub repository
  (50 words)
- *Specification:* This is a list of features that your Vending Machine includes and may be displayed in a bullet point list (100 words)
- *System Flowchart:* A visual depiction of the logical operation of your Vending Machine. An example System Flowchart can be found on Aura. (Creatley can be useful for creating flowcharts https://creately.com/). This may be

accompanied by a short explanation
(50 words).

- *Technical Description:* A description of how the technical aspects of your Vending Machine operates. This should focus on the programming techniques you have used to achieve the final outcome (e.g. if statements, loops, arrays, methods) rather than how a user operates the program (400-500 words)
- *Critical Reflection:* This should describe what aspects of your Vending Machine you find compelling, what could be improved, and what programming skills you need to learn to make such improvements. (300-400 words)
- *Appendix:* A copy of your code should be included in an appendix at the end of your documentation.
  (Not included in word count)

Submission

Please follow the submission instructions below. Work that is submitted incorrectly may not be accepted or could incur a points penalty.

Before submitting have you…

- Checked that any digital work is functioning as expected?
- Spell-checked and grammar-checked any written work that accompanies your digital work? Please make an appointment with the Writing and Learning Centre or speak to your tutor if you are experiencing challenges in this area.
- Formatted your written work to the specification below?
- Referenced all sources of information accurately? Please refer to www.citethemrightonline.com (Harvard) for guidance.

Your development document must be submitted via Turnitin and code submitted via your GitHub repository. Please adhere to the following method:

- Check your vending machine is working as expected.
- Commit and **push** the final version of your code to your Github repository for this assignment (ensure you push all the files required to run your vending machine. In many cases this may just be a single .cpp file).
- Ensure your github profile bio includes your student number
- Copy the link for your unique github repository and include this at the beginning of your development document.
- Save your development document as a Word document.
- Log into Minerva, go to the Assessment tab and submit your finalised document via the appropriate Turnitin Link.

Only code pushed to your Github repository before the assessment deadline will be marked. Ensure you give yourself enough time before your final push. Guides on how to push your code to GitHub are provided on Aura.

Format

All written work must conform to university styling and submission guidelines. They must:

- Be word-processed using a conventional font and size (e.g. Times New Roman, 11 or 12) and 1.5 or double line spaced.
- Contain appropriate in-text citation that supplies an accurate list of references.
- Be accurate in referencing. See Bath Spa guidelines and the Harvard system described at www.citethemrightonline.com.
- Be accurate in spelling and paragraphing.

Marking Criteria

Assignment S2: Utility App will be marked against the following criteria:

1. System Design - 15%
2. Technical Implementation - 50%
3. User Experience - 15%
4. Documentation - 20%

Criteria	Weighting		Marks
System Design	20%	A very limited design that may not be implementable.	0 - 19 (Low Fail)
		A poor system design that omits key features. The vending machine may not be fit for purpose.	20 - 39 (Fail)
		A basic design that meets the minimum requirements stated on the brief.	40 - 49 (Third)
		A fair design that meets the minimum requirements well. Additional features are limited to product categorisation.	50 - 59 (2:2)
		A good design that meets the minimum requirements well and presents some additional features such as product categorisation and secondary purchase options.	60 - 69 (2:1)
		Very good system design that provides several additional features such as product categorisation, secondary purchase options and intelligent purchase suggestion system.	70 - 79 (First)

		Excellent system design that has many additional features including ones beyond those listed on the brief.	80 - 89 (High First)
		Beyond expectations for this level of study.	90 - 100 (Outstanding)
Technical Implementation	40%	A very limited implementation that demonstrates little evidence of programming skill.	0 - 19 (Low Fail)
		A poor implementation that contains errors. Limited programming techniques deployed. May not compile. Coding Conventions have not been observed (comments, indentation, camelCase etc).	20 - 39 (Fail)
		A basic implementation that deploys basic programming techniques. While the program runs some features may not function as expected. Code structure may contain a high level of duplication. Coding Conventions have been observed to a limited degree (comments, indentation, camelCase etc)	40 - 49 (Third)
		A fair implementation that deploys several key programming programming techniques. Largely functional overall with only minor errors. Code suffers from duplication. Coding Conventions have been observed to a fair standard (comments, indentation, camelCase etc), although there is room for improvement	50 - 59 (2:2)

		A good implementation that demonstrates an appropriate range of programming techniques. Code has some minor duplication that could be improved. Coding Conventions have been observed (comments, indentation, camelCase etc), with only minor errors present	60 - 69 (2:1)
		A very good implementation that successfully deploys appropriate programming techniques to minimise code duplication. Very well structured code. Coding Conventions have been observed without errors (comments, indentation, camelCase etc).	70 - 79 (First)
		An excellent implementation that demonstrates several techniques outside the scope of the class. Code structure and functionality is high quality. Coding Conventions have been observed with precision and comments are detailed (comments, indentation, camelCase etc)	80 - 89 (High First)
		Beyond expectations for this level of study.	90 - 100 (Outstanding)
User Experience	15%	A very limited user experience. User instructions are unclear and contain typographical errors. Application is confusing to use.	0 - 19 (Low Fail)
		A poor user experience that presents minimal instructions to the user. Application is navigable but provides erroneous information.	20 - 39 (Fail)

		A basic user experience with limited clarity. May not reveal all necessary information to the user and/or provides ambiguous instructions.	40 - 49 (Third)
		A fair user experience that provides key information but lacks detail and precision (e.g. missing £ signs)	50 - 59 (2:2)
		A good user experience that is accurate throughout. Console outputs are on the whole clear, yet there is room for refinement.	60 - 69 (2:1)
		A very good user experience that provides clear and detailed information to the user. Console is well presented.	70 - 79 (First)
		An excellent user experience that implements techniques beyond those taught in class.	80 - 89 (High First)
		Beyond expectations for this level of study.	90 - 100 (Outstanding)
Documentation	25%	Very limited documentation that demonstrates little or no understanding of the design and build process. Some sections are missing and structure is unacceptable	0 - 19 (Low Fail)
		Poor documentation that provides only a basic understanding of the design and build process. Writing is fragmented and badly structured and/or sections are missing	20 - 39 (Fail)
		Basic documentation that provides key information on the design and build process. Technical implementation lacks correct usage of terminology and may be limited	40 - 49 (Third)

		in depth. Critical reflection provides some limited insights. Structure is acceptable but flowchart may be missing	
		Fair documentation that may lack precision in writing. Design and build stages are discussed to a sound level of detail. Technical implementation only has minor inaccuracies. Critical reflection provides only key insights. Structure is acceptable and all elements are present.	50 - 59 (2:2)
		Good documentation that satisfies the requirements of the brief. Uses accurate technical terminology throughout. Critical reflection provides useful insights but may lack detail on how improvements could be made.	60 - 69 (2:1)
		Very good documentation that satisfies the brief well. Critical reflection is well considered, and the description of technical implementation is well focused. Documentation contains code snippets and screenshots.	70 - 79 (First)
		Excellent documentation that offers a high level of detail and precision. Critical reflection provides valuable insights into the design and development process. Structure and writing is approaching flawless.	80 - 89 (High First)
		Beyond expectations for this level of study.	90 - 100 (Outstanding)

Intended Learning Outcomes (ILOs)

ILO	Assessed
An understanding of the key features of programming including input, output, maths, sequence, iteration and repetition.	
The application of coding conventions to ease the review, maintenance, troubleshooting and debugging of developed software.	✓
The successful implementation of a prototype system that is driven by procedural programming techniques.	✓
An ability to specify applications, discuss their technical implementation and reflect critically on the results.	✓

Mark penalties may be applied to late submissions without prior approval of an extension. Please ensure that you prepare and submit your work in good time to allow for any issues that may arise.