School of Design & Informatics

Assessment Instrument Coversheet

Module Code:	CMP104
Module Title:	Programming with C++
Lecturer:	Dr Suzy Prior and Dr Natalie Coull
Submission Date:	Monday 2nd December 23:59
Feedback Return Date:	Feedback will be provided within 15 working days
Feedback Type: (eg verbal, My Learning Space)	verbal, during the demonstration
Grading Criteria	Refer to Page 7

Submission Requirements:

Your assessment must be submitted via My Learning Space. The maximum file size which can be submitted is 20MB so you may need to reduce the size of any image files within your document.

Guidance on submitting via My Learning Space is available at: http://submit.ac.uk/en_gb/training/student-training/submitting-a-paper, but please contact the Support Enquiry Zone on 01382 308833 or sez@abertay.ac.uk if you have any problems with submitting your work on the My Learning Space.

Submission of your work after the submission date deadline will be deemed as late submission and will incur penalty, including the possibility of the work being awarded a non-submission (NS) grade.

CMP104 Programming with C++

1 Introduction

Module CMP104 Programming with C++ is assessed by a combination of quizzes, practical exercises, and demonstrations of your completed tasks, which show your skills and evidence achievement across the module.

2 Learning Outcomes

All modules at Abertay are defined by module descriptors which are published on Oasis. In each case the module seeks to encourage the student to achieve "Learning Outcomes" and this provides the focus, in particular, of assessments – you need to show that you have achieved them. For CMP104 these outcomes are:

1. Create and run programs written in an object-oriented programming language using an integrated development environment
2. Identify classes and solve routine programming problems
3. Create re-usable software routines in an object-oriented programming language.
4. Utilise standard class library functionality

By the time you have completed the assessed exercises across the term, you will have evidenced all of these outcomes.

3 Assessment

Your assessment for this module comprises two units of assessment. Each unit of assessment is worth 50%. You must evidence all of the learning outcomes in order to pass the module.

Unit 1 (worth 50% of the module) consists of 6 My Learning Space Quizzes, 4 Lab Submissions and a mid semester demo.

- **My Learning Space Quizzes:**
- My Learning Space quizzes will be issued in weeks 3, 4, 5, 6, 7, and 8. Each quiz is worth 10% of the unit grade.

- **Mid Semester Demo**
- You will be asked to demo a lab exercise to a lab tutor in Week 7. You will be asked questions and asked to explain how you achieved a solution to the problem. This is worth 20% of the unit grade.

- **Lab Exercises**
- You will be asked to upload specific exercises from weeks 3, 4, 6 and 8. Each of these is worth 5% of the unit grade.

Your task for **Unit 2** is described over the rest of this document.

4 Assessment Overview - Unit 2 (worth 50% of the module)

Fruit Machine Assignment

For this exercise, we want you to program a simple fruit machine in the console, which the user can play for prizes.

Figure 1 - Real Life Fruit Machines – Credit to Unsplash.com

Your fruit machine should include the following functionality, as a minimum:

- Display 3 columns of characters on the console window
- When the game play begins, the characters in each column should rotate rapidly (i.e. intentionally overprint at the same location)
- The user can choose when to stop the first column from rotating, then the second column, and finally stop the third column.
- If the symbols in all three columns are the same, display a message indicating that the user has hit the jackpot
- If the symbols in only two columns are the same, provide a smaller jackpot prize

You can choose to add additional functionality into your program, limited only by your imagination and programming skills.

To help you get started with the fruit machine assessment, please work through the exercises below, using some existing code that is available of My Learning Space.

Figure 2 shows an example output from previous years – note your submission does not need to look exactly like this, you should design your own output for your submission.

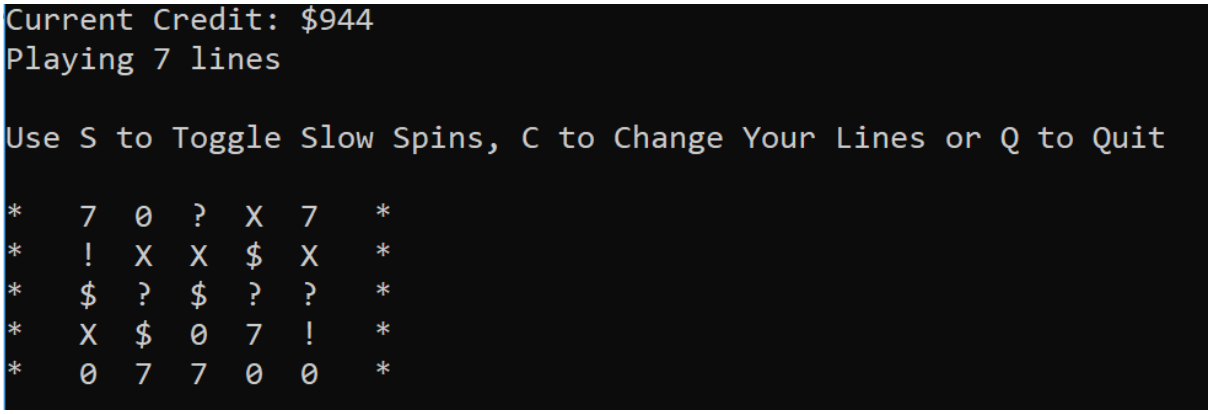

Figure 2 - Example Output

Precursory Task for the Fruit Machine

Copy Andrew LaMothe's code from My Learning Space into a new Visual Studio project.

This program has the appearance of a simple computer game:

- Move the craft left or right (keys 'a', 'A', or 's', 'S' respectively)
- Exit with 'q' or 'Q'.
- Try to avoid the ground fire.

You will see that there is very little functionality included in the game; no scoring, no variety, no levels. We have suggested some ideas below to help you modify this code, this will ultimately help you with the fruit machine exercise. Those coded green are very simple, amber suggests a little more work will be needed, and red signifies more challenging exercises.

1. Craft
    a. The rudimentary craft shape is <-*->. Try to find the line of code that creates it, and see if you can change the shape to something else
2. Colours
    a. Similarly, the colour variations are random and may not necessarily be to your liking. Change the colours to something more to your liking
3. Missiles
    a. Currently the ground fire is represented by a full stop character - but what about using other characters, for example a mixture of characters, or randomly selected characters?
4. Movement
    a. Currently the left and right keys only move the craft one square either way only – why? You could modify this so that the craft moves more than one square with each key press
5. Input
    a. Change the input commands to L and R rather than A and S.
    b. Change the exit character. X for exit would be simple.
    c. And what about moving left and right with the arrow keys? That is trickier still as they are not simple keys at all but a two character pair (cf. http://www.jimprice.com/jim-asc.shtml#keycodes).
6. Speed

a. The Sleep() function takes an integer parameter and waits that number of milliseconds. So 100 in this case means loop ten times per second. Try changing the game speed. (this could be used to control the speed that your fruit machine columns rotate)

7. Displaying a score
    a. declare a new variable to act as a score
    b. include an increment statement in the game loop (while game_running) so that the score increases each time (that will be ten times per second at current speed).
    c. display this number at the top right of the screen. If you were to use the very top line (line 0) your value would scroll off with the spaceship and the two might collide. Supposing that the craft stays on line 0, you could avoid this problem by displaying the score on line 1. Now though you will have to erase the previous score which will have scrolled up to line 0. Alternatively you could allow the two to overlap as long as the game score was over-printing, preferably, in a different colour. (This could be used to control the printing of the symbols for your fruit machine).
    d. introduce some logic to make the game loop stop when your counter reaches a certain value. I suggest you use the value 99 during testing which will give you ten seconds play. Later on you might increase this (999 would give 1 minute 40 seconds).

Think about how you could modify the score display to simulate the rotors of the fruit machine.

Imagine now having three number fields one above the other. As an example, let's say the lower number was 7 and displayed in faint colour (0-7) and the middle number one greater, 8, and displayed in the same colour but bold (8-15) and the upper number similarly one greater, 9, but again faint. Now if these numbers are incremented and over-printed as above we might get the impression of a single rotor of a fruit machine.

Rather than digits you might look to the ASCII characters to see if there might be something better. For example, the playing card suit symbols are there (ASCII 3, 4, 5, and 6). Now aligning three of these triple rotors … you get the idea. Of course to add the logic to determine winnings for two cherries etc. you will need to make use of if statements.

Fruit Machine Submission and Demonstration (worth 50% of module grade)

The solution that you upload to My Learning Space should consist of the following:

- A .cpp file(s) that contains the source code
- Some comments in the source code to explain each section of the code
- A word document providing a short (half page) description of the features that you have successfully implemented and any important design decisions you want to highlight.

How you will be assessed

Your electronic submission to My Learning Space is used to demonstrate your achievements to the moderator and external examiner. Your module tutor will assess your work during the demonstration, where you will be asked some questions on your code. These questions will be straight-forward and simply help us to determine that you understand the code.

You will receive feedback on your program during the demonstration, and the grades will be issued electronically either via email or via My Learning Space once all students have completed their demonstrations. Students who have not uploaded their program to My Learning Space will not be allowed to demonstrate it. **Your attendance at this demonstration is compulsory, failure to attend the demonstration will result in a failing grade being awarded.** The grade will be calculated based on how well your solution meets the criteria, the efficiency of the solution, and your ability to demonstrate an understanding of the code. The grading criteria is provided at the end of this document.

In order to achieve a B or A grade we expect that you will significantly build upon the suggested functionality for your fruit machine.

Grading Criteria

Grade	C++ Knowledge	Programming	Application Features	Report writing and formatting
A/A+	An excellent grasp of knowledge and understanding of C++ syntax and program structure.	Excellent programming quality, following a consistent style and making use of generally accepted best practices during implementation.	An excellent example of a fruit machine with features and functionality well beyond the minimum scope.	Excellent formatting and structure, with clear flow of information between sections. Excellent use of language with correct grammar and spelling.
B/B+	A very good grasp of knowledge and understanding of C++ syntax and program structure.	Very good programming quality, following a consistent style.	A very good example of a fruit machine with features and functionality beyond the minimum scope.	Very good formatting and structure, with good flow of information between sections. Excellent use of language with correct grammar and spelling.
C/C+	A good grasp of knowledge and understanding of C++ syntax and program structure.	Good programming quality, with some minor inconsistencies.	A good example of a fruit machine with features which meet all of minimum scope.	Good formatting and structure with reasonable flow of information between sections.
DD+	A satisfactory grasp of knowledge and understanding of C++ syntax and program structure.	Satisfactory programming quality, with substantial scope for improvement.	A satisfactory example of a fruit machine with features which meet all of minimum scope.	Satisfactory formatting and structure but lacking coherence of sections.
MF	Marginally unsatisfactory knowledge and understanding of C++ syntax.	Marginally unsatisfactory programming quality.	A marginally unsatisfactory example of a fruit machine with features which meet some of minimum scope.	Marginally unsatisfactory formatting and structure.
F	Performance well below the threshold level, with only limited evidence of achievement			
NS	There is no submission, attendance at the demonstration, or the submission contains no relevant material			