CPT109 C Programming and Software Engineering 1 – ASSESSMENT 2

Assessment Number	2
Contribution to Overall Marks	50%
Issue Date	02/11/2020
Submission Deadline	14/12/2020 at 0600 (6am)

Assessment Overview

This assessment aims to test your ability to design a modular program which makes use of files and structures. Application of the software development process (**SDP**) is also under assessment; the five main steps of the software development process are:

1. Problem statement: formulate the problem.
2. Analysis: determine the inputs, outputs, variables, etc
3. Design: define the list of steps (the algorithm) needed to solve the problem.
4. Implementation: the C code has to be submitted as a separate file. Just indicate here the name of the file.
5. Testing: explain how you have tested and verified your C program.

EXERCISE

An entertainment company would like you to develop a version of the popular memory game "Find the Pairs". In this game, a player is shown a grid of randomised pairs of symbols to memorise in a short time. The player then selects pairs of objects to uncover, which remain uncovered if they match. The aim is to turn over all pairs in the smallest number of attempts possible. The program will allow the user to create an account so that their game history can be recorded and viewed each time they login.

Program Requirements

1. Your program should provide users with the ability to create an account. An account should be a structure type variable containing at least: a username, a password and details of their game playing history.
2. All of the accounts should be stored in a data file and accessed by the program.
3. Once a user is logged on to the game they should be able to:
   (i) Start a new game
   (ii) Review their game history
   (iii) Clear their game history
4. Logout
5. Normally, the use of global variables is not allowed. Any use of global variables must be fully justified in your report.
6. The randomised pair grid must be a minimum size of 4 x 4 (i.e. 8 pairs).
7. The grid should initially be displayed to the player for a short time.
8. The player should be able to select pairs of objects to uncover. These remain uncovered if they match and are re-covered if they do not match after a short delay.
9. The total number of guesses it takes to uncover all pairs should be recorded.
10. The game should allow the player to exit to the main menu at any time.

Ideas

Note: the following are only to provide you with ideas of how to implement the required functionality. They do not represent the "best" or only ways to implement the functions.

All player accounts are structure variables and will be stored in a file. The login process can be achieved using a single structure variable, into which each account can be read from the file one by one, each time checking the username until you find the correct players account. The gaming operations can then be performed using that single structure variable. When the player logs out only this one structure needs to be written back to the file for saving in the correct position.

Alternatively, you can create an array of structures in the program and read the whole file into the array, then search the array for the correct users' structure. The game can be played using the correct element in the array of structures. When the player finishes, the whole array can be written to the file.

The randomised grid of symbols can be created using a 2D array.

Try to create your own functions to simplify the programming task.

Consider how your program should function when there is an invalid input.

To make your program more interesting for the user think about the playability. Use your own experience of playing games, for example you might allow different difficulty settings related to the size of the grid which could be changeable, you may record the time it takes to complete the grid, some statistical data about their game history may be of interest.

What should be submitted?

1) A short report (up to a few pages of text plus C source codes) detailing for each question:
   a) **SDP** steps 1 to 3 in the report (Problem statement(10%) + Analysis(10%) + Algorithm Design(10%) + Report Quality(10%)) (40%)
   b) **SDP** step 4 (Implementation + Robustness): C source code including comments. (40%). Do not paste the code into your report just submit the .c source code file and write the file name in your report under the implementation section. This **MUST** compile and run in Visual Studio 2013.
   c) **SDP** step 5 (testing): explain your testing methodology including: what you wanted to test, how you have tested it and the outcome of your tests. (15%). Note: you do not need to include screenshots of all testing results. You can include some to show correct operation and or failures, avoid including pages and pages of screenshots for every test you perform.

   Please refer to the file "CPT109 Marking Guidelines Assignment 2" for the detailed marking scheme.

2) The report in Microsoft Word or pdf format and C source code of your implementation should be zipped into a single file, i.e. the zip file will contain 2 files, one document and one source code.

The naming of Report (.docx or .pdf), Source Code (.c) and Compressed file (.zip or .rar)

- StudentID_LastName_FirstName_AssignmentNumber.docx or .pdf
- StudentID_ AssignmentNumber.c
- StudentID_LastName_FirstName_AssignmentNumber.zip or .rar

For example
- 123456789_Einstein_Albert_2.docx
- 123456789_2.c

Zipped together into:
- 123456789_Einstein_Albert_2.zip

How the work should be submitted?

Should be submitted electronically through the Learning Mall so that the marker can run your programs during marking. Feedback and your grade will also be given through the Learning Mall.

Remember that you are responsible for ensuring that your C code will run in Visual Studio 2013 and to C90 standard and that if it does not without documentary explanation you may get a 0 mark for your implementation.