**ICT283 Revision Exercise**

***James Carppe***

***34827095***

1. **Assumptions (5%)**

*All assumptions made other than those stated in the question that you make about the problem. There will virtually always be assumptions you are implicitly making so think about this very carefully. Also be careful that you do not put in unnecessary assumptions. Assumptions like “The user understands English” or “The user has fingers to type” may appear to make sense but are pointless from the point of view of the program operation. So do not record such assumptions.* ***(5%)***

* The ‘coins.txt’ file’s lines will always be formatted as “[NAME] [AMOUNT] cents in [CURRENCY]”.
  + [NAME] is a string (char \*) no longer than 99 characters.
  + [AMOUNT] is an int that has a value less than 99 – values over 99 will throw a warning and be rounded down.
  + [CURRENCY] is one of three values:
    - US$
    - AU$
    - EUR
* The ‘coins.txt’ file is always present in the executable’s directory.
* The computer has adequate memory to read and compute on the ‘coins.txt’ file’s customers.
* The computer has the correct software to run the executable.

1. **Structure Chart (5%)**

*Structure chart for your program. Show parameter passing.*

1. **Algorithm (20%)**

*Your algorithm written in a uniform fashion using a pseudocode or a similar style and adhering to the conventions required in the unit. Your algorithm should be presented at an appropriate level of detail sufficient to be easily implemented. Submit your high- level algorithm (where necessary) along with algorithms of your decompositions as appropriate to the question.*

*Algorithms that look like the code was written first and then word processed to look like an algorithm would receive no marks.*

<write algorithm here>

….

1. **Test Table (10%)**

*A set of test data in tabular form with expected results and desk check results from your algorithm. Each test data must be justified – reason for selecting that data. No marks will be awarded unless justification for each test data is provided.*

Add rows to the following table as needed. Table can span more than one page. Each test id tests only one condition for the desk check.

For this assignment, there can be up to 10 records in a data file. In the test table below, you might have one test id for 10 records. So the actual 10 records must be in one cell of the test table in the column *Actual data*. Of course there are other test conditions and you need to include those too.

F**aking the outcome of any test will result in no marks given for this entire section. What that means is that if you have a few hundred tests which are fine, but you faked/falsified the outcome of just one, you will get a mark of 0.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test id** | **Test description/justification – what is the test for and why this particular test.** | **Actual data for this test** | **Expected output** | **Actual desk check result when desk check is carried out** | **Desk check outcome – Pass/Fail** |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |

1. **Code (50%)**

*Name and purpose of functions/modules in the source code files. Do not put actual source code here. Code exists as separate source code files that are submitted. Source code files (.c and .h) must be submitted separately and the source code must build (compile and link) to create an executable that operates correctly. Make sure you use the code style required in the unit. No marks awarded if the source code does not build and run.*

|  |  |  |
| --- | --- | --- |
| **File name** | **Name of Functions/modules in the file** | **Purpose of the Function/module** |
| calculator.c | debug\_Change | Debug function used to print the values of a Change struct |
| free\_Change | Free the memory allocated when a Change struct is created |
| calculate\_change | Calculates the change for an amount and a given currency |
| currency.c | currency\_from\_string | Converts a string input into an int representing a currency |
| string\_from\_currency | Converts an int and writes a string to a provided char\*\* pointer |
| currency\_type\_from\_int | Converts an int to a type from a Currency enum |
| customer.c | validate\_amount | Validates an input amount |
| validate\_amount\_limit | Validates an input amount including an upper bound check |
| debug\_Customer | Debug function used to print the values of a Customer struct |
| debug\_Customer\_Cluster | Debug function used to print the values of a Customer\_Cluster struct |
| free\_Customer | Free the memory allocated when a Customer struct is created |
| free\_Customer\_Cluster | Free the memory allocated when a Customer\_Cluster struct is created |
| create\_Customer | Creates a Customer struct and validates the fields |
| create\_Customer\_Cluster | Creates a Customer\_Cluster struct and validates the fields |
| customer\_add\_amount | Adds an amount to an existing Customer structs amount and validates the new value |
| customer\_compare | Compares two Customer structs and returns if they are equal |
| customer\_merge | Merges two Customer structs into one |
| files.c | get\_lines\_in\_file | Gets the number of lines in a FILE\* |
| open\_readonly\_file | Opens a file and returns the opened file inside a ReadOnly\_File struct |
| Open\_readwrite\_file | Opens a file and returns the opened file inside a ReadWrite\_File struct |

|  |  |  |
| --- | --- | --- |
| **File name** | **Name of Functions/modules in the file** | **Purpose of the Function/module** |
|  |  |  |

1. **Results of Program Testing (5%)**

*Results of applying your test data to your final program (tabular form), including a sample printout of your program in operation.*

Add rows to the following table as needed. Table can span more than one page.

Each test id tests only one situation for the test run of the program. Table is copy/paste of the desk check with actual output column showing results of the program output. There should be no duplicated reasons listed in the second column.

F**aking the outcome of any test will result in no marks given for this entire section. What that means is that if you have a few hundred tests which are fine, but you faked/falsified the outcome of just one, you will get a mark of 0.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test id** | **Test description/justification – what is the test for and why this particular test.** | **Actual data for this test** | **Expected output** | **Actual program output when test is carried out** | **Test run outcome – Pass/Fail** |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |

After the above test table, copy/paste sample printouts of your program in operation. You can screen capture and paste here. Make sure you label each printout with the correct *Test id*.

1. **Self-Assessment (5%)**

*Self-assessment of how successful you were in achieving the requirements and a discussion of any problems you encountered.*

Write your self-assessment here. Use as much space as needed. Describe how well your solution meets the requirements. Explain how you can improve your solution. Discuss problems you encountered and how you resolved them.

**Submit a separate file called *evaluation.txt***. This file has two headings, and you enter the required summary as dot points under the headings. The first heading is “**What works**” and the second heading is “**What does not work**”. Do not make any false claims as marks for this component may not be awarded. Testing should be thorough.

The file *evaluation.txt* will also declare if you have checked each submitted file for viruses or malware. Name the tool and version number of the tool that you used to conduct the check. If the checks for viruses/malware are not made and the declaration is not shown in *evaluation.txt*, this assignment will not be marked, and no marks will be given to you. Any delay that results from virus or malware will incur the specified daily penalty for the assignment. Advice on how to do a malware scan is under Unit Info or Essential Resources at the LMS site for this unit.