**ICT159 Assignment 2**

***<Student Name>***

***<Student Number>***

1. **Assumptions**

*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.*

* First assumption :

First of all, the fact that the program has a modular structure and partially includes 2 different file operations, according to the production of the solution algorithm of the problem, made me think more about it. I thought I would have a harder time with those parts.

* Second assumption:

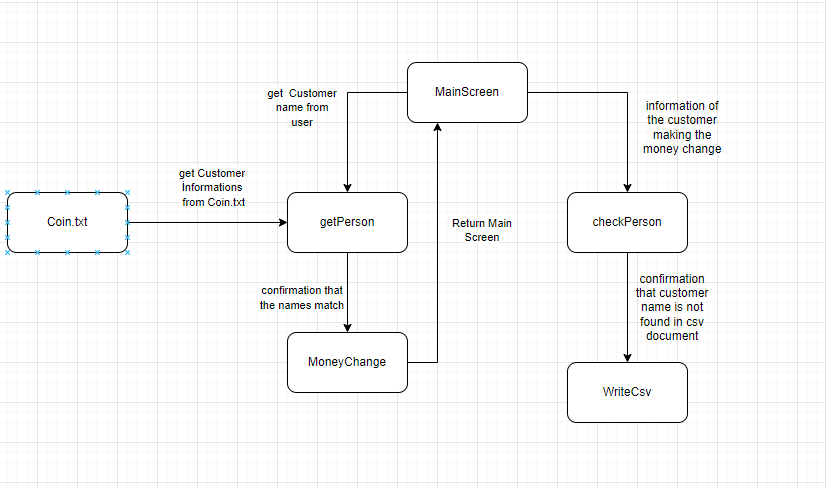
Secondly, I did quite a bit of research on how to do data communication between functions. While I was at the stage of writing the program, I examined the examples from the websites for a long time.

* Third assumption:

I realized that these operations must be different from each other, since we also do write and read operations in Txt and Csv files. One was just a text area while the other contained columns and rows.

1. **Structure Chart**

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



1. **Algorithm**

*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.*

***The Algorithm :***

*The program will proceed by dividing the existing coin from the largest to the smallest, and then subtracting this value to make a coin change. For example, a customer with 90 cents will first divide by 50, subtract 50 cents from 90 cents, and the remainder will be divided by 20, the second largest coin.*

1. **Test Table**

*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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 | Re-entering a name found in the csv file. because Because we don't want to print the same result twice. | Jane  Jane | There İs a same name . You cannot enter this costumer again. | There is a same name in the document. You Can't save again. | **Pass** |
| 2 | Entering a name that does not exist in the text file. because we cannot deal with a customer who does not exist. | Daniels | There is no this name in text file . please write another name. | Name : Daniels  not found.  Please try again. | **Pass** |
| 3 | Entering a number other than 1 and 2 in the menu.  Because Peoples can enter wrong character. We don’t want to close the program. | 3 | You should enter number 1 or 2 | You should enter number 1 or 2 | **Pass** |

1. **Code**

*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, .cpp, and/or .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.*

Extend the following table as needed. Functions/modules need to match what is in the structure chart. If it is the same file name for a number of functions/modules, you write the file name once in the *File name* column for the first function/module listed in the table.

|  |  |  |
| --- | --- | --- |
| **File name** | **Name of Functions/modules in the file** | **Purpose of the Function/module** |
| Coin.c | int MainScreen() | Enter to console Choice Elements |
| Coin.c | int getPerson(char name[20] | Comparison of customer names entered by the user and in the coin.txt document |
| Coin.c | int MoneyChange(int money) | Changing the money and writing to values |
| Coin.c | int WriteCsv() | Writing change.csv document. |
| Coin.c | int checkPerson(char name[20]) | Check are there same name in the change.csv |

1. **Results of Program Testing**

*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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 | Main screen test - is Main screen elements will be true written | - | 1. Enter name  2. Exit | 1. Enter name  2. Exit | Pass |
| 2 | Main screen test – Entering Mistake value | 5 | Your choice is wrong please try again. | You should enter number 1 or 2 | Pass |
| 3 | Main screen test – Choicing quit value | 2 | - | -(The Program has been close)- | Pass |
| 4 | Text File test – will Values be true read | - | Name:Jane  CoinValue:30  Name:Jack  CoinValue:50 | Name:Jane  CoinValue:30  Name:Jack  CoinValue:50 | Pass |
| 5 | Csv File test – Will values be true write | Jack,50,1,0,0,0 | Jack,50,1,0,0,0(But it should be divide per column) | Jack,50,1,0,0,0(But it hasn’t been divide per olumn),  Probably because of my computer . can be tried on another computer. | Fail |

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**

*Self-assessment of how successful you were in achieving the requirements and a discussion of any problems you encountered. This write up is done in this document in the space provided below. You need to also submit a separate file evaluation.txt. A false claim in evaluation.txt would mean that marks for this component would not be awarded. So make sure that you have tested your program thoroughly.*

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.

---

I think the program I wrote has completely solved the problem. It successfully exchanges the client's money and writes it to the csv file. While writing the program, I had difficulties while writing/reading txt and csv files. I did not know how the C program would filter the data in the txt and csv file and pull it into the program. I learned by trial and error. I used a static struct value because it would be difficult to return arrays between functions, and each function would operate on the same array. I think my program has successfully accomplished what it intended.

---

**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.