  
  
**Assignment Cover Sheet**

|  |  |
| --- | --- |
| **Subject Code: CSCI 251** |  |
| **Subject Name: Advanced Programming** |  |
| **Submission Type: Project Submission** |  |
| **Assignment Title: FINAL PROJECT** |  |
| **Student Name: Mohammed Ejazzur** |  |
| **Student Number: 7305849** |  |
| **Lecturer Name: Lim H.C** |  |
| **Due Date: 17/06/2022** |  |
| **Date Submitted: 17/06/2022** |  |
|  | @uowmail.edu.au |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **PLAGIARISM:** The penalty for deliberate plagiarism is FAILURE in the subject. Plagiarism is cheating by using the written ideas or submitted work of someone else. UOWD has a strong policy against plagiarism.  The University of Wollongong in Dubai also endorses a policy of non-discriminatory language practice and presentation.  **PLEASE NOTE:**STUDENTS MUST RETAIN A COPY OF ANY WORK SUBMITTED | **DECLARATION:** I/We certify that this is entirely my/our own work, except where I/we have given fully-documented references to the work of others, and that the material contained in this document has not previously been submitted for assessment in any formal course of study. I/we understand the definition and consequences of plagiarism.  **Signature of Student: Ejazz** |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Optional Marks:** | | **Comments:** | |

https://my.uowdubai.ac.ae/images/scissors.gif

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Lecturer Assignment Receipt**(To be filled in by student and retained by Lecturer upon return of assignment) | | | **Subject:** | **Assignment Title:** | | **Student Name:** | **Student Number:** | | **Due Date:** | **Date Submitted:** | | **Signature of Student:** | | |

https://my.uowdubai.ac.ae/images/scissors.gif

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Student Assignment Receipt** (To be filled in and retained by Student upon submission of assignment) | | | **Subject:** | **Assignment Title:** | | **Student Name:** | **Student Number:** | | **Due Date:** | **Date Submitted:** | | **Signature of Lecturer** | | |

# Introduction

Our final project was to build a simulation program to allow bus administrators to see what the seating allocation of a set of passengers at a set of stops would look like. This would include certain features such as retrieving previous stops for auditing purposes.

I’ve built it both ways, using procedural and Object-Oriented-Programming.

# Summary of Procedural and OOP

Both solutions were built systematically after a general understanding of the program was accomplished and a set of functions and classes were put into project scope.

# Reflections

Better understanding of the C++ programing Language.

A better grasping of how to build solutions to problem statements logically.

How to extract and interpolate data to be readable for a program.

# Body

## Procedural

A screenshot of a computer

Description automatically generated with medium confidence

These were the set of functions used in Procedural part of this solution. There are 3 major functions and 3 variables:

startPlan() | resetPlan() | startAuditProgram()

currentStop, fileReaderPointer, familyCounter

These functions were further split into their own sub-functions to increase code readability and usability:

startPlan: grabListFromFile, seatPasseneger, removePassenger, savePlanToFile

**-startPlan**

This function displays a menu and allows the user to pick an option which will call sub-functions accordingly.

-**grabListFromFile**

The purpose of this function is to read from the passengers.txt file and convert the text from unreadable to something the program can read and build a set of passengers for that specific stop to be added to the simulation.

-**seatPassenger**

This function seats the passenger. Based on the passenger type, it will set the limits to where the passenger may be seated.

-**removePassenger**

This function will comb through the array to find specifically which passenger it will remove. In the case of a family it calls a special function called **removeFamily.**

**-savePlanToFile**

This is a simple function that just saves the bus to a file that can be later accessed for auditing purposes. It is called after each stop has been run.

startAuditProgram: retrievePlan

**-startAuditProgram**

This will display a menu that will allow the user to pick which stop they want to be able to view from records.

**-retrievePlan**

This will retrieve the plan based on the user input and display it.

resetPlan

This is a simple function that sets all global variables and the main array to its default state.

currentStop, fileReaderPointer, familyCounter

-**currentStop**

This is a counter that tracks which stop the bus is in at that stage of the simulation and will increment every time **startPlan() is run.**

**-fileReaderPointer**

Keeps track of where the program stops reading the file after the file reader is no longer reading the passengers for the current stop. This is to prevent the file reader from reading unnecessary lines.

-familyCounter

This is a count of all the families that got on the bus. This is to be used in conjunction with the character ‘F’ to specify that a specific passenger belongs to a family.

It is also used to link two of the children of the family to the mother so during family removal, you just need to search for the mother and the children will be removed too.

## OOP

Diagram

Description automatically generated

# Lessons Learned

As a result of this project, I’ve learnt how to build solutions to problems more systematically and as a step-by-step process to ensure I’m able to cover all possible cases and edge cases for my solutions.