Data Structures and Algorithms

Module Title:	Data Structures and Algorithms
Assignment Type:	Individual Practical Assignment
Project Title:	Visitor List in Immigration Department
Project Date:	October 2019
Assignment Compiler:	Amilcar Aponte
Weighting:	35%
Due Date:	**18th December 2019** Late submissions will be accepted up to 5 days with a penalty of 10% of the grade obtained.
Method of Submission:	**Moodle Uploader** (No email submissions will be allowed. All submissions must be done through the Moodle uploader)

Assignment Introduction

The Immigration department has contacted you outlining that they currently do not have a method of processing who is next in line to be seen by the immigration officer at the counter. As the immigration department is quite busy in the months of September and February due to new students in the country, often people take a ticket, leave and then come back later to see if it is their time to meet with the immigration officer.

Sometimes, some people need special treatment due to special conditions, that need to jump into the middle of the list of waiting people.

Design and implement a piece of software that allows the immigration officer in the immigration department to add new candidates to a queue, check the position of a person by id in the queue, and add or remove people from the queue at different positions depending on their priority status.

Use a Linked List based queue data structure to store the data.

Specific Requirements

- This assignment is focused upon the utilisation of a **Doubly Linked List** data storage solution as underlying structure for a queue.
- You have to implement a program to allow a staff member at the counter to add a new person into the queue. When a new person is added into the system, they will be required to add their first name, last name, date of arrival, passport number and priority level. When the person is added, after its information is collected, they should be added as a new object according to their priority level.

- At any time, the staff member should have the ability to see what position in the queue a person is, by typing in a unique ID number that is given to the person when they register in the system. Keep in mind that the ID does not necessarily correspond to the position of the person in the queue.

- There should be three priority levels:
    - High Priority
    - Medium Priority
    - Low Priority

- When a low priority person has been added to the queue, they must go straight to the end of it.

- When a high priority person has been added to the queue, they must go straight to the start of the queue, unless the first person is of high priority, in such case they should be added after the last high priority person in the queue.

- When a medium priority person has been added to the queue, the must go after all the high priority people, but before all the low priority people in the queue.

- At any time, the staff member should have the ability to delete a person from the system by entering in their unique ID number. If the person is removed from the queue, their object should be removed and whoever was in front of them should be jointed to the person who was behind them.

- For each of the operations which are being performed on the queue, individual methods should be created to encapsulate the functionality.

- A method should exist to cut off the last **N numbe**r of records from the queue. If the staff member types in 3. Then the last 3 objects on the linked list should be removed.

- Given a person unique number, the staff member should be able to update the information for that person, without impacting where they currently are in the queue.

Extra marks

If you would like to achieve a distinction, consider to add some extra layers of functionality, such as, but not limited to:

- Create dynamically some testing data
- Implementing data persistency through an external database
- Implementation of a Graphic User Interface

Notes

- Comment your code!!

- All code must be your own and not cut and paste from somewhere on the internet.

- This is an individual assignment, if your assignment is similar to another student's assignment you will both score zero.

- **No email submissions will be allowed. All submissions must be done through the Moodle uploader.**
- **Late submissions will be accepted up to 5 days with a penalty of 10% of the grade obtained.**

Deliverables

- Source code for your application, uploaded to Moodle

Marking Scheme Summary

Description	Weighting
The underlying data structure for the queue is a doubly linked list. Individual methods exist for each of the functionality available to the doubly linked list, encapsulating the process. Code is well structured and commented.	15
The user can add a new person to the queue and it gets automatically organised according to its priority. Code is well structured and commented.	20
The user can remove the first person from the queue when this has been looked after. Code is well structured and commented.	5
The user can check to see what position in the queue a person currently is. Code is well structured and commented.	10
A person in any position can be deleted from the queue, connecting the person who was in front of them to the person who was behind them.	5
The user can delete N number of records from the end of the queue.	10
Can update information for a single person, without impacting their position in the list.	5
Extra functionality has been implemented. Code is well structured and commented.	30

TOTAL	**100**