Data Structures and Algorithms

Module Code:

Module Title:	Data Structures and Algorithms
Assignment Type:	Assignment
Project Title:	Visitor List in Immigration Department
Project Date:	18/12/2017
Assignment Compiler:	Dr. Muhammad Iqbal
Weighting:	30%
Due Date:	**18**th December 2017
Method of Submission:	Moodle Uploader

Assignment Introduction

An 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 the people take a ticket, leave and then come back later to see if it is their time to be meet with the immigration officer. Since the various kinds of visitors (Business, Patients and Families) are present in the immigration department for the extension of their immigration. Sometimes, the parents with their kids needs some special treatment because of children less than three years, that need to jump into the middle of the list of waiting people for their immigration renewal.

Design and implement a command line application that allows the immigration officer in the immigration department to add new candidates to a list, check the position of a current person by name in the list, and add or remove people from the list at different positions.

Use a Linked List data structure to store the data.

Specific Requirements

- This assignment is focused upon the utilisation of a **Linked List** data storage solution.
- A simple command line interface should be created, which will allow a staff member at the counter to add a new person into the list. When a new person is added into the system, they will be required to add their first name, last name, date of arrival and passport number. When the person is added, after its information is collected, they should be added as a new object to the end of the Linked List.
- At any time, the staff member should have the ability to see what position in the linked list a person is, by typing in a unique ID number that is given to the person when they register in the system.
- A function should exist to select a position in the linked list, e.g., position 4, and put a new person with a kid into that position. The person who was at position 4, should then be pushed to position 5 and so on for each different person.

- If a person comes into the immigration department who has kids (age less than 1 year), they should be given the very first position in the linked list. The person who was originally number 1, should then be moved to position 2.

- 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 linked list, 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 list, 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 linked list. 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 list.

Notes

- All code must be your own, and not taken off the internet or from a friend.

- All code should be documented, outlining what each function is doing.

Deliverables

- Source code for your application, uploaded to Moodle

Marking Scheme Summary

Description	Weighting
1- Can check to see what number in a linked list a person currently is.	10
2- Can add a new person to end the of the linked list	10
3- A method should exist to insert a position into a specific position into the linked list.	10
4- A person can be deleted from the linked list, connecting the person who was in front of them to the person who was behind them.	10
5- A person can be added to the top of the linked list, moving the first person who was there to the second position in the list.	10
6- Individual methods exist for each of the functionality available to the linked list, encapsulating the process.	10
7- Ability to delete N number of records from the end of the linked list.	10
8- Can update information for a single person, without impacting their position in the list.	10
9- Overall code quality and structure.	20
TOTAL	**100**