|  |  |  |
| --- | --- | --- |
| American University of Sharjah  College of Engineering  Department of Computer Science & Engineering  P. O. Box 26666, Sharjah, UAE |  | **Instructors:** Dr. Michel Pasquier  **Lab Instructor:** Praveena Kolli  **Office:** EB2-126  **Phone**: 971-6-5152352  **e-mail**: pkolli@aus.edu  **Semester**: Spring 2021 |

**CMP 305 L – Data Structures and Algorithms**

**Lab. Assignment 3 – Singly Linked List**

***Objectives:***

* To understand singly linked lists and their implementation
* To program functions for processing singly linked lists

**Note:**

***Lab:*** Exercises 1 and 2 (10 marks)

***Bonus and optional*:** Exercise 3 (1 mark)

**Exercise1:**

You are required to implement the following functions for *singly linked lists.*

Note that codes for creating a singly linked list from an array of values is provided. Also, testing code is given to make sure that all functions work as expected.

1. find: returns true if the given value is found in the list, else false.

template <typename Object>

bool findInSLL(SingleNode<Object>\* head, Object value);

1. insertAfter: finds a given value and inserts a new node with the new value after it. Returns true if the insertion is successful, else false.

template <typename Object>

bool insertAfterSLL(SingleNode<Object>\*& head, Object givenValue, Object newValue)

1. erase: find a given value and delete the node from the linked list. Returns true if the deletion is successful, else false.

template <typename Object>

bool eraseSLL(SingleNode<Object>\*& head, Object givenValue)

Sample output:

**Printing Singly linked list:**

**11 22 44 77 66 88**

**Inserting 55 after 44 :**

**11 22 44 55 77 66 88**

**Inserting 99 after 88 (inserting at the end of the list):**

**11 22 44 55 77 66 88 99**

**Deleting 77:**

**11 22 44 55 66 88 99**

**Deleting 99 (deleting last element in the list):**

**11 22 44 55 66 88**

**Deleting 11 (deleting first element in the list):**

**22 44 55 66 88**

**Deleting -99 (trying to delete an element which is not present in the list):**

**Value not Found!**

**Exercise2:**

You are to develop a simple *queue* system for a customer service counter, based on singly linked lists.

Customers are processed “first come, first served” from the front of the queue, while new customers join at the end of the queue. (This is known as FIFO or “First In First Out”.)

Implement the queue system using a singly liked list. You need to have both head and tail pointers, to the first node (front) and last node (end), respectively, in order to manage the queue.

Your program should display a menu for the user to choose an option, as follows:

* 1. Add a customer to the queue
  2. Serve the next waiting customer
  3. Print all customers in the queue
  4. Exit

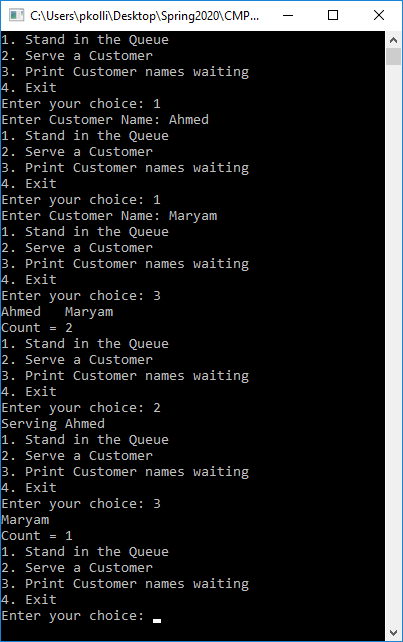
For options #1, your program should ask the user to enter a customer name (string) and add it at the end of the queue.

Option #2, print customer’s name to be served and delete the customer from queue.

Option #3, print customer names waiting in the queue along with the count.

Option #4, exit the program.

Sample input and output:



\*You may use provided main function in driver.cpp file to continue.

**Bonus and optional**

**Exercise3:**

implement the following functions for *singly linked lists,*

1. insertBefore: find a given value and insert a node before it with a new value. Returns true if the insertion is successful, else false.

template <typename Object>

bool insertBeforeSLL(SNode<Object>\*& head, Object givenValue, Object newValue);

1. deleteDuplicates: deletes duplicates of a given value by leaving the first occurrence. Returns true if the duplicate deletion is successful, else false.

template <typename Object>

bool deleteDuplicatesSLL(SNode<Object>\*& head, Object givenValue);

1. deleteAllDuplicates: deletes all duplicates by leaving the first occurrences only.

template <typename Object>

void deleteAllDuplicatesSLL(SNode<Object>\*& head);

Note: make sure your logic is efficient and less complex.

Sample output:

**Testing Bonus :**

**Inserting 33 before 44 :**

**22 33 44 55 66 88**

**Inserting 0 before 22 (inserting at the start of the list):**

**0 22 33 44 55 66 88**

**Adding duplicates:**

**44 0 22 33 44 44 44 55 66 88**

**Remove duplicates of 44:**

**44 0 22 33 55 66 88**

**Adding duplicates:**

**44 11 0 88 22 33 55 66 88 11 88**

**Remove all duplicates:**

**44 11 0 88 22 33 55 66**