|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE200L: Data Structures & Algorithms Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 27/09/2022** |
| **Teaching Assistant: Muhammad Sufyan Ashraf** | **Semester: Fall 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 5A. Basic Operations on Linked Lists - I**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| NIMRA MAQBOOL | BSCE21012 |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to provide the knowledge of basic data structures and their implementations.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

The variable that is used to hold the memory address of another variable is called a **pointer** variable or simply a pointer. The data type of the variable (whose address a pointer is to hold) and the pointer variable must be the same. A pointer variable is declared by placing a asterisk (\*) after data type or before variable name in data type statement. E.g. if pointer variable “p” is to hold memory address of an integer variable it is declared as:

**int \*p;**

or to hold address of a float type variable we can declare as:

**float \*rep;**

Like arrays, **Linked** **List** is a linear data structure. Unlike arrays, linked list elements are not stored at a contiguous location; the elements are linked using pointers. They include a series of connected nodes. Here, each node stores the data and the address of the next node.

**Templates** are a feature of the C++ programming language that allows functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one.

**Lab Task**

**Task A**

You need to continue using the code produced in last lab.

**// Add function to insert the node at start in linked list**

void insertNodeAtStart (T)

{

}

**// Add function to insert the node at end in linked list**

void insertNodeAtEnd (T)

{

}

**// Add function to print linked list**

void printLinkedList ()

{

}

|  |
| --- |
| // Paste your code here  **FUNCTION.H:**  #include <iostream>  using namespace std;  template<typename T> class Node { //made a class of node public:  T data; //declared data in public  Node \*nextPtr;  Node(T val) {  data = val; //made a constructor to set values  nextPtr = NULL;  }   void setNextPtr(Node \*n) {  nextPtr = n; //set the next ptr  }   Node \*getNextPtr() {  return nextPtr; //get the value of next ptr  } }; template <class T > class linkList { //making 1 other class of linklist public:  Node<T> \*tail;  Node<T> \*head; //made some pointers  linkList() {  head = nullptr; //declared them to zero  tail = nullptr;  }  void insertElementInTheStart(int val) {  auto \*temp = new Node<T>(val); //given the memory  temp->data=val;  temp->nextPtr=nullptr;  if (head == NULL) {  head = temp; //checking and then putting the value  tail = temp;  } else {  tail->nextPtr = temp;  tail = temp;  }  }  void display()  {  Node<T> \*temp=head; //declaring  while(temp!=NULL)  {  cout<<temp->data<<"\t";  temp=temp->nextPtr; //storing  }  }  void insertElementInTheEnd(int newElement) {  Node<T>\* temp = new Node<T>(newElement); //declaring  temp->data = newElement;  temp->nextPtr = nullptr;  if(head == nullptr) {  head = temp;  } else {  Node<T>\* temp1 = head;  while(temp1->nextPtr!= nullptr)  temp1 = temp1->nextPtr;  temp1->nextPtr = temp;  }  } };  // // Created by Lenovo on 9/27/2022. // #include <iostream> #include "Functions.h"  using namespace std;  int main() {  Node<int> n(5);  n.setNextPtr(0);  n.getNextPtr();  linkList <int> l;  int opt1;  cout<<"CHOOSE OPTIONS."<<endl;  cout<<"1.ADD AN ELEMENT AT THE START OF THE LIST."<<endl;  cout<<"2.ADD AN ELEMENT AT THE LAST OF THE LIST."<<endl;  cout<<"3.EXIT."<<endl;  cin>>opt1;  int opt2;  if(opt1==1){  int val;  do{  cout<<"\nYOU WANT TO ADD MORE THAN 1 VALUE?\nFOR 1 ENTER 0.\nFOR MORE THAN 1 ENTER 1.\nIF YOU WANT TO STOP ADDING VALUES ENTER 2."<<endl;  cin>>opt2;  if(opt2==1){  cout<<"\nENTER THE VALUE YOU WANT TO ADD."<<endl;  cin>>val;  l.insertElementInTheStart(val);  l.display();  }  if(opt2==0){  cout<<"\nENTER THE VALUE YOU WANT TO ADD."<<endl;  cin>>val;  l.insertElementInTheStart(val);  cout<<"THE 1 VALUE IN LIST IS = ";  l.display();  }  if(opt2==2){  cout<<"THE FINAL ARRAY IS = ";  l.display();  }  if(opt2!=1 && opt2!=0 && opt2!=2){  cout<<"YOU HAVE ENTERED AN INVALID ARGUMENT"<<endl;  }  }while(opt2==1);   }  if(opt1==2){  int newElement;  int opt;  cout<<"ENTER ELEMENT TO PUT AT THE END OF THE ARRAY = ";  cin>>newElement;  l.insertElementInTheEnd(newElement);  cout<<endl;  do{  cout<<"\nDO YOU WANT TO ADD MORE ELEMENTS?"<<"\nENTER 1 FOR YES AND 0 FOR NO." <<endl;  cin>>opt;  if(opt==1){  l.display();  cout<<endl;  cout<<"\nENTER ELEMENT TO PUT AT THE END OF THE ARRAY = ";  cin>>newElement;  cout<<endl;  l.insertElementInTheEnd(newElement);  l.display();  }  if(opt==0){  cout<<"THE FINAL LIST IS = ";  l.display();  exit(3);  }  }while(opt==1);  }  if(opt1==3){  cout<<"YOU CHOOSE TO EXIT."<<endl;  exit(3);  } }  // Paste your output here  Text  Description automatically generated  Text  Description automatically generated  Text  Description automatically generated |

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & Github Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_