Assignment #02

Name :

**M.ANSAR**

Class :

**3rd semester**

Roll No:

**21011556-078**

Course Title :

**Data Structure &Algorithm**

Course Code :

**IT-223**

Course Instructor :

**Sir Azib Mehmood**

Department :

**Information Technology**



**Operations on Linked List**

//Linked list implementation

#include “iostream”

#include “string”

Using namespace std;

//The class “node” refers to a node in linklist

Class node {

Public:

Int data; //to store data

Node\* next; //to store address of next node

Node(int value){

Data=value;//initialization

Next=NULL;//initialization

}

};

Class linkedlist{

Private:

Node\* head;//to store address of first node

Public:

Linkedlist(): head(NULL){}

//Function to insert values in Linked List

Void insert(int value){

//A new node with given value is created

Node\* newnode= new node(value);

If(!head){

//If there is no head i.e NULL, the newly created node is the head

Head = newnode;

}else{

//Otherwise, the list is traversed and the new node is appended at the end of the list

Node\* temp=head;

While(temp→next){

Temp=temp→next;

}

Temp→next=newnode;

}

}

//Function to display the elements of the list

Void display(){

Node\* temp=head;

Cout≪endl;

While(temp){

Cout≪temp→data≪” “≪endl;

Temp=temp→next;

}

Cout≪endl;

}

Void delstart(){

//create a new node and assign it value of head

Node\* newnode=head;

//Change the value of head to next node

Head=head→next;

//Delete the first node

Delete newnode;

Cout≪”Element deleted.”≪endl;

}

Void delend() {

//New node created and assigned the value of head

Node\* temp = head;

//Check whether the second last is null and stop the loop

While (temp→next→next != NULL) {

Temp = temp→next;

}

//now temp points to the second-to-last node

Delete temp→next;

Temp→next = NULL; // Set the next of the second-to-last node to NULL

Cout≪”Element deleted.”≪endl;

}

Void delnth(int n){

//Again, create a new node and assign it value of head

Node\* temp = head;

//To check whether the given number of element exists in list

If (temp == NULL || temp→next == NULL) {

Cout ≪ “Invalid position” ≪ endl;

Return;

}

//FOR Loop to reach the required number of element

For(int i=1; i<n-1 && temp != NULL; i++){

Temp=temp→next;

}

//Create a new node and refer it the value of node to be deleted

Node\* todelete=temp→next;

//replace the deleted node by the node next to it

Temp→next = todelete→next;

Delete todelete;

Cout≪”Element deleted.”≪endl;

}

Void insertnth(int n, int e){

//Creating a new node with value “e”

Node\* newnode = new node(e);

//Again, create a new node and assign it value of head

Node\* temp = head;

//To check whether the given number of element exists in list

If (temp == NULL || temp→next == NULL) {

Cout ≪ “Invalid position” ≪ endl;

Return;

}

//FOR Loop to reach the required number of element

For(int i=1; i<n-1 && temp != NULL; i++){

Temp=temp→next;

}

// Check if position is invalid

If (temp == NULL) {

Cout ≪ “Invalid position” ≪ endl;

Delete newnode; // Delete the new node to avoid memory leak

Return;

}

// Insert the new node in between

Newnode→next = temp→next;

Temp→next = newnode;

Cout≪”Element Inserted.”≪endl;

}

Void updatenth(int n, int e){

//Again, create a new node and assign it value of head

Node\* temp = head;

//To check whether the given number of element exists in list

If (temp == NULL || temp→next == NULL) {

Cout ≪ “Invalid position” ≪ endl;

Return;

}

//FOR Loop to reach the required number of element

For(int i=1; i<n-1 && temp != NULL; i++){

Temp=temp→next;

}

//Assign the value given by user to the element

Temp→next→data=e;

Cout≪”Element updated.”≪endl;

}

Void search(int n){

//Temporaray pointer given value of head

Node\* temp=head;

//Bool to break the code if value is found

Bool found = false;

//Integer to keep track of index

Int x=0;

//Loop for traversal through the list

While (temp != NULL) {

If (temp→data == n) {

Cout ≪ “Element exists at NO.”≪x+1≪” in the list.”≪ endl;

Found = true;

Break;

}

Temp = temp→next;

X++;

} //If element is not found

If (!found) {

Cout ≪ “Element does not exist.” ≪ endl;

}

}

Void searchandupdate(int a){

//Calling the existing function to search

Linkedlist∷search(a);

//Taking new value of element

Cout≪”Enter the new value of the element: “≪endl;

Int val;

Cin≫val;

//Temporary pointer

Node\* temp=head;

Bool found = false;

//Traversal through list

While (temp != NULL) {

If (temp→data == a) {

//Assignment of new value

Temp→data=val;

Cout≪”Value of element is updated.”≪endl;

Found = true;

Break;

}

Temp = temp→next;

}

}

};

Int main()

{

//object creation

Linkedlist mylist;

//insertion of elements

Mylist.insert(1);

Mylist.insert(2);

Mylist.insert(3);

Mylist.insert(4);

Mylist.insert(5);

Mylist.insert(6);

Mylist.insert(7);

Mylist.insert(8);

//displaying the elements

Mylist.display();

//Menu to perform operations

Int choice;

Do{

Cout≪”Enter operation to perform:\n1)Delete from beginning \n2)Delete from end \n3)Delete from nth position \n4)Insert at nth position \n5)Search \n6)Update at nth position \n7)Search and update \n8)Display \n0)Exit”≪endl;

Cin≫choice;

Switch(choice){

Case 1:

Mylist.delstart();

Break;

Case 2:

Mylist.delend();

Break;

Case 3:

Cout≪”Enter which element do you want to delete:”≪endl;

Int n;

Cin≫n;

Mylist.delnth(n);

Break;

Case 4:

Int a,b;

Cout≪”Enter the position where you want to insert a new element:”≪endl;

Cin≫a;

Cout≪”Enter the value of element:”≪endl;

Cin≫b;

Mylist.insertnth(a,b);

Break;

Case 5:

Int search;

Cout≪”Enter element you want to search:”≪endl;

Cin≫search;

Mylist.search(search);

Break;

Case 6:

Int m,e;

Cout≪”Enter which element do you want to update:”≪endl;

Cin≫n;

Cout≪”Enter the value of element:”≪endl;

Cin≫e;

Mylist.updatenth(m,e);

Break;

Case 7:

Int search2;

Cout≪”Enter element you want to search:”≪endl;

Cin≫search2;

Mylist.searchandupdate(search2);

Break;

Case 8:

Mylist.display();

Break;

Case 0:

Return 0;

Break;

Default:

Cout≪”Invalid selection. Please enter 0-7 as input.”;

}

}while(choice!=0);

Return 0;

}