LAB WORK - 1

Aawishkar Tiwari

Computer Engineering – 3rd Semester

Roll no – 59

GitHub : https://github.com/Aawishkar/DSA_LAB.git

Linked List

Linkedlist.h

```h
#ifndef LinkedList_h
#define LinkedList_h

class Node{
    public:
        int data;
        Node *next;
        Node(){}
        Node(int x){
            data =x;
            next=nullptr;
        }
        Node(int x, Node *y){
            data =x;
            next=y;
        }
};
class LinkedList{
    public:
        Node *HEAD;
        Node *TAIL;

        LinkedList(Node *HEAD =nullptr, Node *TAIL=nullptr){
            this->HEAD=HEAD;
            this->TAIL=TAIL;
        }

        bool isEmpty();
        void addToHead(int data);
        void addToTail(int data);
        void add(int data, Node * predecessor);
        void removeFromHead();
        void remove(int data);
        Node* retrieve(int data,Node *);
        bool search(int data);
        void traverse();
};

#endif
```

Linkedlist.cpp

```cpp
#include<iostream>
#include "linkedlist.h"

using namespace std;


//Checking List is empty or not
bool LinkedList::isEmpty(){
    if(HEAD==nullptr){
        return true;
    }
    else{
        return false;
    }
}

//Adding data to head of list
void LinkedList::addToHead(int data){
    Node* newNode= new Node(data);
    if(isEmpty()){
        HEAD=newNode;
        TAIL=newNode;
   }
   else{
        newNode->next=HEAD;
        HEAD=newNode;
   }
}

//Adding data to tail of list
void LinkedList::addToTail(int data){
    Node *newNode= new Node(data);
    if(isEmpty()){
        HEAD=newNode;
        TAIL=newNode;
    }
    else{
        TAIL->next=newNode;
        TAIL=newNode;
    }
}
```

```
41  }
42
43  //Adding data in the list
44  void LinkedList::add(int data, Node* predecessor){
45      Node *newNode = new Node(data);
46      newNode->next=predecessor->next;
47      predecessor->next=newNode;
48  }
49
50  //Removing data from the head of list
51  void LinkedList::removeFromHead(){
52      Node *nodeToDelete =HEAD;
53      HEAD=HEAD->next;
54      delete nodeToDelete;
55
56  }
57
58  //Removing data from the list
59  void LinkedList::remove(int data){
60     if (data==HEAD->data){
61          removeFromHead();
62     }
63     else{
64          Node *prev, *temp;
65          prev=HEAD;
66          temp=HEAD->next;
67
68          while(temp!=nullptr){
69              if(temp->data=data){
70                  break;
71              }
72              else{
73                  prev=prev->next;
74                  temp=temp->next;
75              }
76          }
77          if (temp!=nullptr){
78              prev->next=temp->next;
79              delete temp;
80              if (prev->next==nullptr){
81                  TAIL=prev;
82              }
```

```
            }
        }

    }
}


//Retrieve data from the Linked list
Node *LinkedList::retrieve(int data, Node *outputNodePointer)
{
    Node *temp = outputNodePointer;
    while (temp != nullptr)
    {
        if (temp->data == data)
        {
            return (temp);
        }
        temp = temp->next;
    }
    return nullptr ;
}

//Searching from LinkedList
bool LinkedList::search(int data)
{
    Node *temp = HEAD;
    while (temp->next != NULL)
    {
        if (temp->data == data)
        {
            return true;
        }
    temp = temp->next;

    }

    return false;
}

//Traverse in the Linked List
void LinkedList::traverse(){
    Node *temp;
```

```
    Node *temp;
    temp=HEAD;
    cout<<"\nContents of Linked List are";
    while(temp!=nullptr){
        cout<<"\t"<<temp->data;
        temp=temp->next;
    }
}
```

main.cpp

```cpp
#include<iostream>
#include "linkedlist.h"

using namespace std;
int main(){
    LinkedList list;
    Node *ptr;

    if(list.isEmpty()){
        cout<<"\nThe list Is Empty";
    }
    else{
        cout<<"\nThe List is Not Empty";
    }

    list.addToHead(12);
    cout<<"\n12 is successfully added to head";

    list.addToTail(88);
    cout<<"\n88 is successfully added to tail ";

    list.traverse();

    list.addToHead(66);
    cout<<"\n66 is successfully added to head";

    list.addToTail(11);
    cout<<"\n11 is successfully added to tail ";

    list.traverse();

    list.add(21,list.HEAD);
    cout<<"\n21 has been added successfully";

    list.traverse();

    list.removeFromHead();
    cout<<"\n66 is successfully removed from head";
    list.traverse();

    list.remove(11);
    cout<<"\n11 has been successfully removed";
```

```cpp
    list.traverse();

    cout<<"\nRetrieved Data: "<<list.retrieve(11,list.HEAD);

    cout<<"\nSearch Result"<<endl;
    cout<<list.search(11)<<endl;
    cout<<list.search(21)<<endl;

}
```

Output Screen

```
PS C:\DSA_LAB\LinkedList>
>> g++ main.cpp ./src/linkedlist.cpp -I include
PS C:\DSA_LAB\LinkedList> ./a.exe

The list Is Empty
12 is successfully added to head
88 is successfully added to tail
Contents of Linked List are     12      88
66 is successfully added to head
11 is successfully added to tail
Contents of Linked List are     66      12      88      11
21 has been added successfully
Contents of Linked List are     66      21      12      88      11
66 is successfully removed from head
Contents of Linked List are     21      12      88      11
11 has been successfully removed
Contents of Linked List are     21      88      11
Retrieved Data: 0x1a1ac0
Search Result
0
1
PS C:\DSA_LAB\LinkedList>
```