DSA Assignment 5

Name: Jashnoor Singh
Roll Number: 1024030106

Answer 1:

```
// develop a menu driven program for different functions to be performed on linked list

#include<iostream>
using namespace std;

class Node{
    public:
    int data;
    static int total_nodes;
    Node* next;
    Node(int data){
        this->next = NULL;
        this->data = data;
        total_nodes++;
    }
};
void print(Node* &head){
    Node* temp=head;
    while(temp!=NULL){
        cout<<temp->data<<" ";
        temp=temp->next;
    }
}
void InsertAtHead(Node* &head, int data){
    Node* temp=new Node(data);
    temp->next=head;
    head=temp;
}
void InsertionAtTail(Node* &tail, int data){
    Node* temp = new Node(data);
    tail->next=temp;
    tail=temp;
}
void InsertAfterCertainNode(int num, int data, Node* &head){
    Node* temp = head; // making a new node which points to head
    if(num>Node::total_nodes){
        cout<<"Enter a valid number that is in range \n";
        return;
    }
    int count=1;
    while(count!=num){
        temp=temp->next;
        count++;
    }
    Node* value = new Node(data);
    value->next=temp->next;
    temp->next=value;

}
void DeleteBegining(Node* &head){
```

```
    if (head==NULL){
        cout<<"List is empty ";
        return;
    }
    Node* temp=head; // making a temporary node to point to head
    head=head->next;
    delete temp; // deleting the first node
    Node::total_nodes--; // decreasing the value
}
void DeleteEnd(Node* &head, Node* &tail){
    if(head==NULL){
        cout<<"List is empty";
        return;
    }
    Node* temp=head;
    int num=1;
    while(num!=(Node::total_nodes)-1){
        temp=temp->next;
        num++;
    }
    tail=temp;
    temp=temp->next;
    tail->next=NULL;
    delete temp;
    Node::total_nodes--;
}
void DeleteSpecific(Node* &head, int value){
    if(value<=0 || value>Node::total_nodes){
        cout<<"Enter a valid node index ";
        return;
    }
    Node* temp=head;
    int num=1;
    while(num!=value-1){
        temp=temp->next;
        num++;
    }
    Node* newtemp=temp->next;
    temp->next=newtemp->next;
    delete newtemp;
}
void SearchNode(Node* &head, int value){
    Node* temp= head;
    int count=1;
    while(temp->data != value){
        temp=temp->next;
        count++;
        if(count==Node::total_nodes){
            cout<<"Number not present ";
            return;
        }
    }
    cout<<"Number is present at "<<count<<" index";
    return;
}
int Node::total_nodes=0;

int main(){
    int number;
    int value;
```

```
    int index;
    Node *node1=new Node(5); // we are making a one node linked list with value 5 to start the
process
    Node *head=node1;
    Node *tail=node1;
    cout<<"Press 1 to insert element at begining \nPress 2 to insert at end\nPress 3 to insert at
some index\nPress 4 to delete from begining\nPress 5 to delete from end\nPress 6 to delete at
some specific index\nPress 7 to search for some node and display its position from head if it is
present\nPress 8 to print the linked list\nPress 9 to exit the program";
    while(number!=9){
    cout<<"\nEnter your choice\n";
    cin>>number;
    switch(number){
        case 1:
        cout<<"Enter the value to insert in begining \n";
        cin>>value;
        InsertAtHead(head,value);
        break;

        case 2:
        cout<<"Enter value to insert in end \n";
        cin>>value;
        InsertionAtTail(tail, value);
        break;

        case 3:
        cout<<"Enter value to insert \n";
        cin>>value;
        cout<<"Enter index to insert \n";
        cin>>index;
        InsertAfterCertainNode(index,value,head);
        break;

        case 4:
        DeleteBegining(head);
        break;

        case 5:
        DeleteEnd(head,tail);
        break;

        case 6:
        cout<<"Enter index you want to delete"<<endl;
        cin>>index;
        DeleteSpecific(head,index);

        case 7:
        cout<<"Enter value you want to search"<<endl;
        cin>>value;
        SearchNode(head,value);
        break;

        case 8:
        cout<<"Linked list is "<<endl;
        print(head);
        break;

        case 9:
        exit(0);
```

```
        default:
        cout<<"Enter a valid number ";
    }
    }
}
```

Output

```
Press 1 to insert element at begining
Press 2 to insert at end
Press 3 to insert at some index
Press 4 to delete from begining
Press 5 to delete from end
Press 6 to delete at some specific index
Press 7 to search for some node and display its position from head if it is present
Press 8 to print the linked list
Press 9 to exit the program
Enter your choice
1
Enter the value to insert in begining
2

Enter your choice
2
Enter value to insert in end
5

Enter your choice
3
Enter value to insert
2
Enter index to insert
2

Enter your choice
8
Linked list is
2 5 2 5
Enter your choice
4

Enter your choice
5

Enter your choice
1
Enter the value to insert in begining
10

Enter your choice
2
 Enter value to insert in end
 20

 Enter your choice
 6
 Enter index you want to delete
 2

 Enter your choice
 7
 Enter value you want to search
 10
 Number is present at 1 index
 Enter your choice
 8
 Linked list is
 10 2 20
 Enter your choice
 9
jashnoorsingh@Jashnoors-MacBook-Air DSA assignment5 %
```

Answer 2

```
#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* next;
    static int total_nodes;
    Node(int data) {
```

```
        this->data = data;
        this->next = NULL;
        total_nodes++;
    }
};
int Node::total_nodes=0;
void print(Node* head){
    while (head != NULL){
        cout<<head->data<<" ";
        head=head->next;
    }
    cout<<endl;
}
void InsertEnd(Node* &tail, int data) {
    Node* temp=new Node(data);
    tail->next=temp;
    tail=temp;
}
void deleteOccurrences(Node* &head, int key) {
    while (head != NULL && head->data == key){
        Node* temp=head;
        head=head->next;
        delete temp;
        Node::total_nodes--;
    }
    Node* curr = head;
    while (curr != NULL && curr->next != NULL) {
        if (curr->next->data == key){
            Node* temp=curr->next;
            curr->next=curr->next->next;
            delete temp;
            Node::total_nodes--;
        } else{
            curr=curr->next;
        }
    }
}
int main() {
    Node* n1=new Node(1);
    Node* tail=n1;
    Node* head=n1;

    InsertEnd(tail, 2);
    InsertEnd(tail, 2);
    InsertEnd(tail, 3);
    InsertEnd(tail, 1);
    InsertEnd(tail, 1);
    InsertEnd(tail, 5);
    InsertEnd(tail, 6);
    cout<<"Original list: ";
    print(head);
    int key;
    cout<<"Enter key value: ";
    cin>>key;
    deleteOccurrences(head,key);
    cout<<"Modified list: ";
    print(head);
    cout<<"Total nodes left: "<< Node::total_nodes<<endl;
}
```

Output

```
Original list: 1 2 2 3 1 1 5 6
Enter key value: 1
Modified list: 2 2 3 5 6
Total nodes left: 5
jashnoorsingh@Jashnoors-MacBook-Air DSA assignm
 assignment5/" && g++ q2.cpp -o q2 && "/Users/j
Original list: 1 2 2 3 1 1 5 6
Enter key value: 2
Modified list: 1 3 1 1 5 6
Total nodes left: 6
jashnoorsingh@Jashnoors-MacBook-Air DSA assignm
 assignment5/" && g++ q2.cpp -o q2 && "/Users/j
Original list: 1 2 2 3 1 1 5 6
Enter key value: 3
Modified list: 1 2 2 1 1 5 6
Total nodes left: 7
jashnoorsingh@Jashnoors-MacBook-Air DSA assignm
 assignment5/" && g++ q2.cpp -o q2 && "/Users/j
Original list: 1 2 2 3 1 1 5 6
Enter key value: 5
Modified list: 1 2 2 3 1 1 6
Total nodes left: 7
jashnoorsingh@Jashnoors-MacBook-Air DSA assignm
 assignment5/" && g++ q2.cpp -o q2 && "/Users/j
Original list: 1 2 2 3 1 1 5 6
Enter key value: 6
Modified list: 1 2 2 3 1 1 5
Total nodes left: 7
jashnoorsingh@Jashnoors-MacBook-Air DSA assignm
 assignment5/" && g++ q2.cpp -o q2 && "/Users/j
Original list: 1 2 2 3 1 1 5 6
Enter key value: 10
Modified list: 1 2 2 3 1 1 5 6
Total nodes left: 8
```

Answer 3:

```
// middle of linked list
#include <iostream>
using namespace std;
class Node{
    public:
    int data;
    static int total_nodes;
    Node* next;

    Node(int data){
        this->data = data;
        this->next = NULL;
        total_nodes++;
    }
};
int Node::total_nodes=0;
void insertAtEnd(Node* &tail,int data){
    Node* temp=new Node(data);
    tail->next=temp;
    tail=temp;
}
void middle(Node* head){
    int num=1;
    while(num!=((Node::total_nodes)/2)+1){
        head=head->next;
```

```
        num++;
    }
    cout<<"Middle node is "<<head->data;
}
int main(){
    Node *n1=new Node(1);
    Node* tail=n1;
    Node* head=n1;
    insertAtEnd(tail,5);
    insertAtEnd(tail,4);
    insertAtEnd(tail,3);
    insertAtEnd(tail,2);
    if((Node::total_nodes)%2==0){
        cout<<"we have even number of nodes and middle element is not there ";
    }
    else{
        middle(head);
    }
}
```

Output

```
Middle node is 4%
```

Answer 4:

```
#include <iostream>
using namespace std;
class Node{
    public:
    int data;
    Node* next;

    Node(int data){
        this->data = data;
        this->next = NULL;
    }
};
void insertAtBegining(Node* &head, int data){
    Node* temp= new Node(data);
    temp->next=head;
    head=temp;
}
void insertAtEnd(Node* &tail,int data){
    Node* temp=new Node(data);
    tail->next=temp;
    tail=temp;
}
void print(Node* &newhead){
    Node* temp=newhead;
    while(temp!=NULL){
        cout<<temp->data<<" ";
        temp=temp->next;
    }
}
```

```
Node* reversal(Node* head) {
    Node* newhead = NULL;
    Node* temp = head;
    while (temp != NULL) {
        insertAtBegining(newhead, temp->data);
        temp = temp->next;
    }
    return newhead;
}

int main(){
    Node* n1= new Node(1);
    Node *tail=n1;
    Node *head=n1;
    insertAtEnd(tail,2);
    insertAtEnd(tail,3);
    insertAtEnd(tail,4);
    insertAtEnd(tail,5);
    cout << "Original list: ";
    print(head);
    Node* reversed = reversal(head);
    cout << "\nReversed List: ";
    print(reversed);
}
```

Output

```
Original list: 1 2 3 4 5
Reversed List: 5 4 3 2 1 %
```