DSA Assignment 5

Q1

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

struct Node {
   int data;
   Node* next;
};

void InsertAtBeginning(Node*&head, int value){
   Node*newNode=new Node();
   newNode->data=value;
   newNode->next=head;
   head=newNode;
}

void InsertAtEnd(Node*&head, int value){
   Node*newNode= new Node();
   newNode->data=value;
   newNode->next=NULL;

   if(head==NULL){
      head=newNode;
      return;
```

```
    }
    Node*temp= head;
    while(temp->next!=NULL){
        temp=temp->next;
    }
    temp->next=newNode;

}
void InsertAtBetween(Node*&head, int value, int position){
    Node*newNode=new Node();
    newNode->data=value;
    newNode->next=NULL;

    if (position==0){
        newNode->next=head;
        head=newNode;
        return;
    }

    Node*temp=head;
    for (int i=0; i<position-1 && temp!=NULL; i++){
        temp=temp->next;
    }

    if (temp == NULL) {
        cout << "Position out of range!" << endl;
        delete newNode;
        return;
```

```
    }
    newNode->next=temp->next;
    temp->next=newNode;
}

void DeleteAtBeginning(Node*&head){
    if (head == NULL) {
        cout << "List is empty, nothing to delete." << endl;
        return;
    }
    head = head->next;
}
void DeleteAtEnd(Node*&head){
    if (head == NULL) {
        cout << "List is empty, nothing to delete." << endl;
        return;
    }
    if(head->next==NULL){
        delete head;
        head=NULL;
        return;

    }

    Node*temp=head;
    Node*temp2;
    while(temp->next!=NULL){
        temp2=temp;
```

```
        temp=temp->next;
    }
    delete temp;
    temp2->next=NULL;



}
void DeleteAtBetween(Node*&head, int position){
    if (head == NULL) {
        cout << "List is empty, nothing to delete." << endl;
        return;
    }
    if (position == 1) {
        head = head->next;
        return;
    }
    Node*temp=head;
    for(int i=0; i<position-2 && temp->next != NULL;i++){
        temp=temp->next;
    }
    Node*temp2=temp;
    temp=temp->next;
    temp2->next=temp->next;
    delete temp;

}
int searchNode(Node*&head, int value){
    Node*temp=head;
```

```
    int count=1;
    while (temp != NULL) {
        if (temp->data == value) {
            return count;
        }
        temp = temp->next;
        count++;
    }
    return -1;
}
void displayNode(Node*&head){
    Node*temp=head;
    while(temp!=NULL){
        cout<<temp->data<<"->";
        temp=temp->next;
    }

}



int main(){

    Node*head =NULL;
    int value,num=0;

    while(num!=9){
        cout<<"What do you want to proceed with: \n";
        cout<<"1. Insertion at the beginning.\n";
```

```
cout<<"2. Insertion at the end.\n";
cout<<"3. Insertion in between\n";
cout<<"4. Deletion from the beginning.\n";
cout<<"5. Deletion from the end.\n";
cout<<"6. Deletion of a specific node\n";
cout<<"7. Search for a node and display its position from head.\n";
cout<<"8. Display all the node values.\n";
cout<<"9.Exit";
cin>>num;
int tempNum;
int tempPos;
switch (num)
{
case 1:
   cout<<"Enter the number to insert: ";
   cin>>tempNum;
   InsertAtBeginning(head,tempNum);
   break;
case 2:
   cout<<"Enter the number to insert: ";
   cin>>tempNum;
   InsertAtEnd(head,tempNum);
   break;
case 3:
   cout<<"Enter the number to insert: ";
   cin>>tempNum;
   cout<<"Which position to insert";
   cin>>tempPos;
```

```
    InsertAtBetween(head,tempNum,tempPos);
    break;
case 4:
    DeleteAtBeginning(head);
    break;
case 5:
    DeleteAtEnd(head);
    break;
case 6:
    cout<<"Which position to delete";
    cin>>tempPos;
    DeleteAtBetween(head,tempPos);
    break;
case 7:
    cout<<"Which number to find";
    cin>>tempNum;
    tempPos=searchNode(head,tempNum);
    if(tempPos!=-1) {
        cout<<tempNum<<" found at position "<<tempPos<<endl;
    }
    else{
        cout<<tempNum<<"not found in the list."<<endl;
    }
    break;
case 8:
    displayNode(head);
    break;
case 9:
```

```
            cout<<"Exiting!";
            break;

        default:
        cout<<"Choose a valid option!";
            break;
        }
    }


    return 0;
}
```

```
What do you want to proceed with:
1. Insertion at the beginning.
2. Insertion at the end.
3. Insertion in between
4. Deletion from the beginning.
5. Deletion from the end.
6. Deletion of a specific node
7. Search for a node and display its position from head.
8. Display all the node values.
9.Exit1
Enter the number to insert: 46
What do you want to proceed with:
1. Insertion at the beginning.
2. Insertion at the end.
3. Insertion in between
4. Deletion from the beginning.
5. Deletion from the end.
6. Deletion of a specific node
7. Search for a node and display its position from head.
8. Display all the node values.
9.Exit8
46->What do you want to proceed with:
```

Q2

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

struct Node {
   int data;
   Node* next;
};
```

```
int main(){
    int num=0;
    while(num<=0){
        cout<<"Enter number of elements in the list: ";
        cin>>num;
    }

    Node* head=NULL;
    int value;

    for(int i=0;i<num;i++){
        cout<<"Element number "<<i+1<<" : ";
        cin>>value;
        Node*temp=new Node;
        temp->data=value;
        temp->next=NULL;
        if (head == NULL) {
            head = temp;
        }
        else {
            Node*current=head;
            while(current->next != NULL){
                current=current->next;
            }
            current->next=temp;
        }
    }
```

```
int key;
cout<<"Enter the key to count and delete: ";
cin>>key;

Node*current =head;
Node*prev =NULL;
int count=0;

while(current!=NULL){
   if(current->data==key){
      count++;
      Node* toDelete =current;
      if(prev==NULL){
         head=current->next;
      }
      else{
         prev->next=current->next;
      }
      current=current->next;
      delete toDelete;
   }
   else{
      prev=current;
      current=current->next;
   }
}

cout<<"Count: "<<count<< endl;
```

```
    cout<<"Updated Linked List: ";
    current = head;

    while(current != NULL){
        cout<<current->data;
        if (current->next != NULL) cout << "->";
        current = current->next;
    }
    cout << endl;

    return 0;
}
```

```
Enter number of elements in the list: 4
Element number 1 : 52
Element number 2 : 14
Element number 3 : 64
Element number 4 : 22
Enter the key to count and delete: 52
Count: 1
Updated Linked List: 14->64->22


=== Code Execution Successful ===
```

Q3

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

struct Node{
   int data;
   Node* next;
};

void findMiddle(Node*&head){
   Node*temp=head;
   int count=0;
   while(temp!=NULL){
      temp=temp->next;
      count++;
   }
   temp=head;
   for(int i=0;i<count/2;i++){
      temp=temp->next;

   }
   cout<<"Middle value is: "<<temp->data<<endl;
}

int main(){

   int num=0;
   while(num%2==0){
      cout<<"Enter number of elements in list: ";
```

```
    cin>>num;
}
Node*head=NULL;

int arr[num];
int value=0;
cout<<"Enter the elements: ";
for(int i=0;i<num;i++){
    cout<<"Element number "<<i+1<<" : ";
    cin>>value;
    Node*temp=new Node;
    temp->data=value;
    temp->next=NULL;
    if (head == NULL) {
        head = temp;
    }
    else {
        Node*current=head;
        while(current->next != NULL){
            current=current->next;
        }
        current->next=temp;
    }
}
findMiddle(head);



return 0;
```

}

```
Enter number of elements in list: 5
Enter the elements: Element number 1 : 62
Element number 2 : 75
Element number 3 : 12
Element number 4 : 36
Element number 5 : 82
Middle value is: 12
```

Q4

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

struct Node {
   int data;
   Node* next;
};

int main(){
   int num=0;
   while(num <= 0){
      cout<<"Enter number of elements in the list: ";
      cin>>num;
   }
```

```
Node* head =NULL;
int value;

for(int i=0;i<num;i++){
   cout<<"Element number "<<i+1<<" : ";
   cin>>value;
   Node*temp=new Node;
   temp->data=value;
   temp->next=NULL;
   if (head==NULL) {
      head= temp;
   }
   else {
      Node*current=head;
      while(current->next != NULL){
         current=current->next;
      }
      current->next=temp;
   }
}

Node*prev= NULL;
Node*current= head;
Node*next= NULL;

while(current!=NULL){
   next=current->next;
   current->next= prev;
```

```
        prev=current;
        current=next;
    }
    head = prev;

    cout<< "Reversed Linked List: ";
    current=head;
    while(current != NULL){
        cout<<current->data;
        if (current->next != NULL) cout<<"->";
        current = current->next;
    }
    cout << "->NULL" << endl;

    return 0;
}
```