Assignment-2

1. Develop a menu driven program for the following operations of on a Singly Linked List.

(a) Insertion at the beginning.

(b) Insertion at the end.

(c) Insertion in between (before or after a node having a specific value, say 'Insert a new Node 35 before/after the Node 30').

(d) Deletion from the beginning.

(e) Deletion from the end.

(f) Deletion of a specific node, say 'Delete Node 60').

(g) Search for a node and display its position from head.

(h) Display all the node values.

Ans 1)

```
#include <iostream>

using namespace std;



struct Node{

int data;

Node* next;

}*start=NULL,*ptr=NULL,*newptr=NULL, *rear=NULL;


class single_llist

{

   public:

       Node* CreateElement(int);

       int count(Node*);

       void Insert_begin(Node *);
```

```
        void Insert_end(Node*);

        void Insert_after_a_node(Node*);

        void Deletion_front();

        void Deletion_end();

        void Deletion_specific_node();

        void search();

        void Display(Node*);

        single_llist()

        {

            start = NULL;

        }

};


Node* single_llist::CreateElement(int no)

{

newptr= new Node;

if(newptr==NULL)

{

cout<<"Memory could not be alloted";

Display(start);

exit(-1);

}

else

{

newptr->data=no;

newptr->next=NULL;

}

return newptr;

}
```

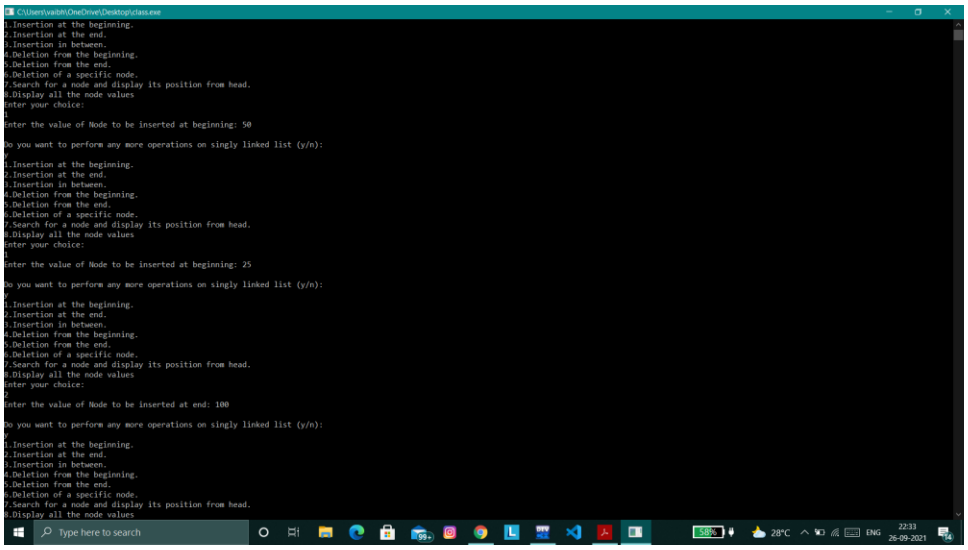
```
void single_llist::Insert_begin(Node* ptr)
{
if(start==NULL)
{
start=ptr;
rear=ptr;
}
else
{
ptr->next=start;
start=ptr;
}
}


void single_llist::Insert_end(Node* p)
{
if(start==NULL)
start=rear=p;
else
{
rear->next=p;
rear=p;
}
}
void single_llist::Insert_after_a_node(struct Node* temp)
{
int value;
```

```
    struct Node *ptr=start, *prev;
    cout<<"\nEnter the value of the previous node: ";
    cin>>value;
    if(start)
    {
            while(ptr&&ptr->data!=value)
            ptr=ptr->next;
            if(ptr)
            {
            temp->next=ptr->next;
            ptr->next=temp;
            }
            else{
                    cout<<"Entered value does not exist in list!";
                    return;
 }
}
else
cout<<"List is empty! Try insertion at front or end.";
}


void single_llist::Deletion_front()
{
if(start==NULL)
{
cout<<"Underflow";
exit(-1);
}
```

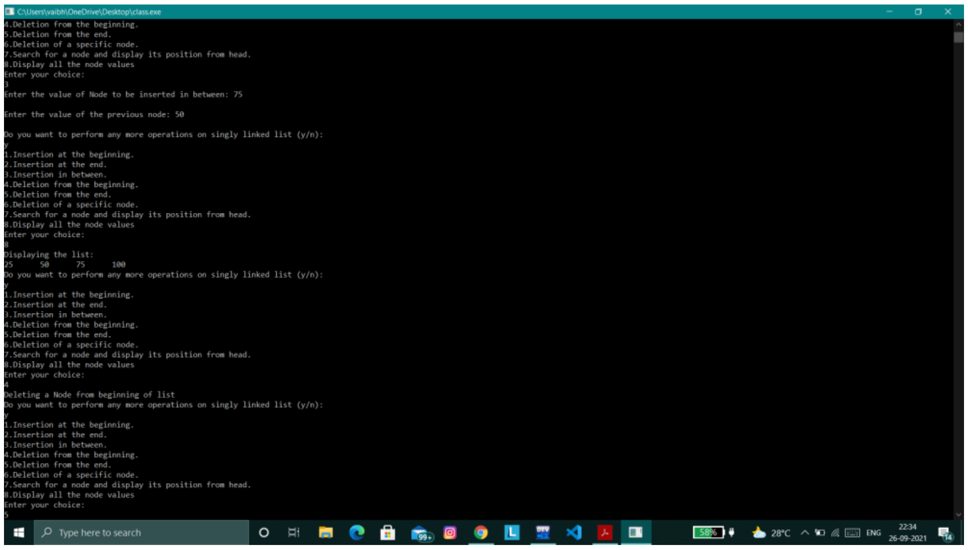
```
else if(start==rear)

{

Node* p=start;

start=rear=NULL;

delete p;

}

else{

Node* p=start;

start=start->next;

delete p;

}

}


void single_llist::Deletion_end()

{

struct Node* ptr=start;

if(start==NULL)

{

cout<<"Underflow!!";

exit(-2);

}

else if(start==rear)

{

start=rear=NULL;

delete ptr;

}

else{

while(ptr->next!=rear)

ptr=ptr->next;
ptr->next=NULL;
delete rear;
rear=ptr;
}
}

void single_llist::Display(Node* link)
{
cout<<"\n";
while(link!=NULL)
{
cout<<link->data<<"\t";
link=link->next;
}
}

void single_llist::search()
{
  int value, pos = 0;
  bool flag = false;
  if (start == NULL)
  {
    cout<<"List is empty"<<endl;
    return;
  }
  cout<<"\nEnter the value to be searched: ";
  cin>>value;
  struct Node *s;
  s = start;
```

```
    while (s != NULL)
    {
        pos++;
        if (s->data == value)
        {
            flag = true;
            cout<<"Element "<<value<<" is found at position "<<pos<<endl;
            break;
        }
        s = s->next;
    }
    if (!flag)
        cout<<"Element "<<value<<" not found in the list"<<endl;
}

void single_llist::Deletion_specific_node()
{
int value;
Node *ptr=start, *prev=start;
cout<<"\nEnter the value of node to be deleted:";
cin>>value;
if(start)
{
while(ptr&&ptr->data!=value)
{
prev=ptr;
ptr=ptr->next;
}
if(ptr==NULL)
cout<<"Value does not exist in list";
else if(ptr==start)//value found at start of list
Deletion_front();
else
{
prev->next=ptr->next;
delete ptr;
cout<<"Node is successfully deleted!";
}
}
else{
cout<<"List is empty! Try Insertion at beginning or end.";
}

}

void printMenu(){
cout<<"1.Insertion at the beginning.\n2.Insertion at the end.\n3.Insertion in between.\n";
cout<<"4.Deletion from the beginning.\n5.Deletion from the end.\n6.Deletion of a specific node.\n";
cout<<"7.Search for a node and display its position from head.\n8.Display all the node values\n";
cout<<"Enter your choice:"<<endl;
}


int main()
{
single_llist list;
int no,n;
int ch;
```

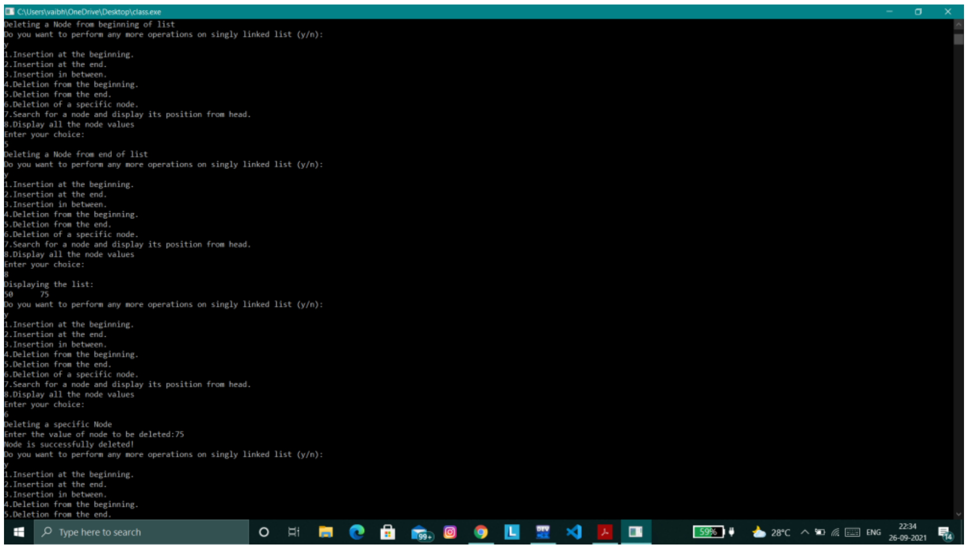
```
char reply;


do
{
printMenu();
cin>>ch;
switch(ch)
{
case 1:
cout<<"Enter the value of Node to be inserted at beginning: ";
cin>>no;
list.Insert_begin(list.CreateElement(no));
break;
case 2:
cout<<"Enter the value of Node to be inserted at end: ";
cin>>no;
list.Insert_end(list.CreateElement(no));
break;
case 3:
cout<<"Enter the value of Node to be inserted in between: ";
cin>>no;
list.Insert_after_a_node(list.CreateElement(no));
break;
case 4:
cout<<"Deleting a Node from beginning of list";
list.Deletion_front();
break;
case 5:
cout<<"Deleting a Node from end of list";
list.Deletion_end();

break;

case 6:

cout<<"Deleting a specific Node";

list.Deletion_specific_node();

break;

case 7:

cout<<"Searching a Node and displaying its position from head";

list.search();

break;

case 8:

cout<<"Displaying the list:";

list.Display(start);

break;

default:

cout<<"Invalid choice:";

}

cout<<"\nDo you want to perform any more operations on singly linked list (y/n):"<<endl;

cin>>reply;

}while(reply=='y'||reply=='Y');

}
```



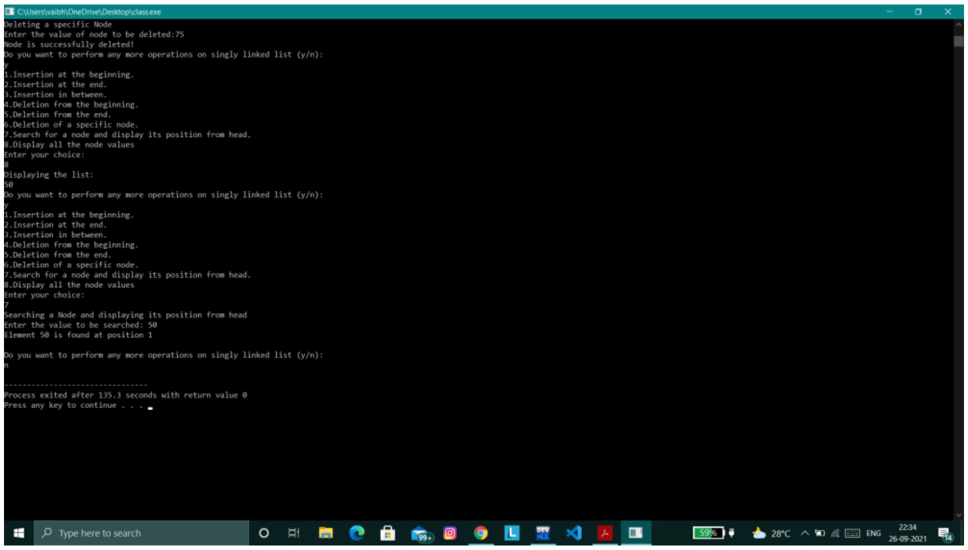
```
1.Insertion at the beginning.
2.Insertion at the end.
3.Insertion in between.
4.Deletion from the beginning.
5.Deletion from the end.
6.Deletion of a specific node.
7.Search for a node and display its position from head.
8.Display all the node values
Enter your choice:
1
Enter the value of Node to be inserted at beginning: 50

Do you want to perform any more operations on singly linked list (y/n):
y
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
Enter your choice:
1
Enter the value of Node to be inserted at beginning: 25

Do you want to perform any more operations on singly linked list (y/n):
y
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
Enter your choice:
2
Enter the value of Node to be inserted at end: 100

Do you want to perform any more operations on singly linked list (y/n):
y
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
```



```
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
Enter your choice:
3
Enter the value of Node to be inserted in between: 75

Enter the value of the previous node: 50

Do you want to perform any more operations on singly linked list (y/n):
y
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
Enter your choice:
8
Displaying the list:
25      50      75      100
Do you want to perform any more operations on singly linked list (y/n):
y
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
Enter your choice:
4
Deleting a Node from beginning of list
Do you want to perform any more operations on singly linked list (y/n):
y
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
Enter your choice:
5
```



```
C:\Users\vaibh\OneDrive\Desktop\class.exe
Deleting a Node from beginning of list
Do you want to perform any more operations on singly linked list (y/n):
y
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
Enter your choice:
5
Deleting a Node from end of list
Do you want to perform any more operations on singly linked list (y/n):
y
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
Enter your choice:
8
Displaying the list:
50      75
Do you want to perform any more operations on singly linked list (y/n):
y
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
Enter your choice:
6
Deleting a specific Node
Enter the value of node to be deleted:75
Node is successfully deleted!
Do you want to perform any more operations on singly linked list (y/n):
y
1.Insertion at the beginning.
2.Insertion at the end.
3.Insertion in between.
4.Deletion from the beginning.
5.Deletion from the end.
```



```
C:\Users\vaibh\OneDrive\Desktop\class.exe
Deleting a specific Node
Enter the value of node to be deleted:75
Node is successfully deleted!
Do you want to perform any more operations on singly linked list (y/n):
y
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
Enter your choice:
8
Displaying the list:
50
Do you want to perform any more operations on singly linked list (y/n):
y
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
Enter your choice:
7
Searching a Node and displaying its position from head
Enter the value to be searched: 50
Element 50 is found at position 1

Do you want to perform any more operations on singly linked list (y/n):
n

--------------------------------
Process exited after 135.3 seconds with return value 0
Press any key to continue . . .
```

Q2) Write a program to count the number of occurrences of a given key in a singly linked list and then delete all the occurrences. For example, if given linked list is 1->2->1->2->1->3->1 and given key is 1, then output should be 4. After deletion of all the occurrences of 1, the linked list is 2->2->3.

Ans:

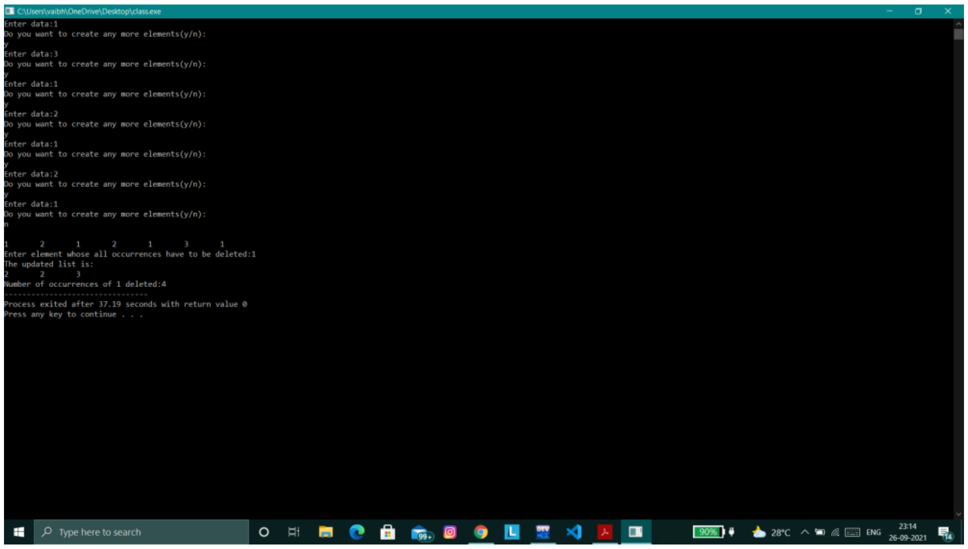
```
#include<iostream>

using namespace std;

struct Node{

int data;
```

```
Node* next;
}*start=NULL,*ptr=NULL, *newptr=NULL, *rear=NULL;

void Display(struct Node*);

Node* CreateElement(int no)
{
newptr= new Node;
if(newptr==NULL)
{
cout<<"Memory could not be alloted";
Display(start);
exit(-1);
}
else{
newptr->data=no;
newptr->next=NULL;
}
return newptr;
}

void Insert_begin(Node* ptr)
{
if(start==NULL){
start=ptr;
rear=ptr;
}
else
{
```

```
ptr->next=start;

start=ptr;

}

}



void Deletion_front()

{

if(start==NULL)

{

cout<<"Underflow";

exit(-2);

}

else if(start==rear)

{

Node* p=start;

start=rear=NULL;

delete p;

}

else{

Node* p=start;

start=start->next;

delete p;

}

}


void Delete_all_occurrences(int key)

{

int n=0;
```

```
Node* temp=start,*prev=start, *Free=NULL;
if(!start)
{
cout<<"List is empty";
return;
}
else{
while(temp)
{
if(temp==start&&temp->data==key)
{
Deletion_front();//start moves to next element after deletion at front
temp=start;
n++;
continue;
}
else if(temp->data==key)
{
prev->next=temp->next;
Free=temp;
temp=temp->next;
delete Free;
n++;
}
else
{
prev=temp;
temp=temp->next;
}
}

}

cout<<"The updated list is:";

Display(start);

cout<<"\nNumber of occurrences of "<<key<<" deleted:"<<n;

}


void Display(Node* link)

{

cout<<"\n";

while(link!=NULL)

{

cout<<link->data<<"\t";

link=link->next;

}

}


int main()

{

int no;

int key;

char reply;

cout<<"Create a linked list by inserting at front:\n"<<endl;

do

{

cout<<"Enter data:";

cin>>no;

ptr=CreateElement(no);

Insert_begin(ptr);
```

```
cout<<"Do you want to create any more elements(y/n):"<<endl;

cin>>reply;

}while(reply=='y'||reply=='Y');

Display(start);

cout<<"\nEnter element whose all occurrences have to be deleted:";

cin>>key;

Delete_all_occurrences(key);

}
```



```
C:\Users\vaibh\OneDrive\Desktop\class.exe
Enter data:1
Do you want to create any more elements(y/n):
y
Enter data:3
Do you want to create any more elements(y/n):
y
Enter data:1
Do you want to create any more elements(y/n):
y
Enter data:2
Do you want to create any more elements(y/n):
y
Enter data:1
Do you want to create any more elements(y/n):
y
Enter data:2
Do you want to create any more elements(y/n):
y
Enter data:1
Do you want to create any more elements(y/n):
n

1       2       1       2       1       3       1
Enter element whose all occurrences have to be deleted:1
The updated list is:
2       2       3
Number of occurrences of 1 deleted:4
--------------------------------
Process exited after 37.19 seconds with return value 0
Press any key to continue . . .
```