Question:

Write a menu driven program that maintains a singly linked list whose elements are stored in on ascending order and implements the following operations (using separate functions):

a) Insert a new element

b) Delete an existing element

c) Find the locations of a given element

d) Display all the elements

CODE:

```
#include<iostream>

#include<cstdio>

#include<cstdlib>

using namespace std;
```

```
struct node

{

int info; struct node *next;
}*start;


class single_llist

{

public:

node* create_node(int);

void insert_pos();

void delete_pos();

void sort();
void search();
```

```
void display();
void single_list(){

start = NULL;

}

};

int main()

{

int choice, nodes, element, position, i;

single_llist sl;

start = NULL;

{
```

```
while (1)
{
cout<<"-----------------------"<<endl;
cout<<"Operations on singly linked list"<<endl;
cout<<"-----------------------"<<endl;

cout<<"1.Insert node at position"<<endl;

cout<<"2.Delete an existing element"<<endl;

cout<<"3.locations of a given element"<<endl;
cout<<"4.Display Linked List"<<endl;

cout<<"5.Exit "<<endl;
cout<<"Enter your choice:";
cin>>choice;
switch(choice)
{
   case 1:
 cout<<"Inserting Node at a given position:"<<endl;
 sl.insert_pos();
 cout<<endl;
```

```
 break;
case 2:
 cout<<"Delete a particular node: "<<endl;
 sl.delete_pos();
 break;

case 3:
 cout<<"Search element in Link List: "<<endl;
 sl.search();
 cout<<endl;
 break;

case 4:
 cout<<"Display elements of link list"<<endl;
 sl.display();
 cout<<endl;
 break;

 case 5:
 cout<<"Exiting..."<<endl;
 exit(1);
 break;
```

```
 default:
 cout<<"Wrong choice"<<endl;
}
}
}
}

node *single_llist::create_node(int value)
{
 struct node *temp, *s;
 temp = new(struct node);
 if (temp == NULL)
   {
 cout<<"Memory not allocated "<<endl;
 return 0;
 }

else
 {
 temp->info = value;
 temp->next = NULL;
 return temp;
```

```
 }
}

void single_llist::insert_pos()
{
 int value, pos, counter = 0;
 cout<<"Enter the value to be inserted: ";
 cin>>value;
 struct node *temp, *s, *ptr;
 temp = create_node(value);
 cout<<"Enter the postion at which node to be inserted: ";
 cin>>pos;
int i;
 s = start;
 while (s != NULL)
 {
 s = s->next;
 counter++;
 }
if (pos == 1)
{
 if (start == NULL)
```

```
{
start = temp;
start->next = NULL;
}
else
{
ptr = start;
start = temp;
start->next = ptr;
}
}
else if (pos > 1 && pos <= counter)
{
s = start;
for (i = 1; i < pos; i++)
{
ptr = s;
s = s->next;
}
ptr->next = temp;
temp->next = s;
}
```

```
else
 {
 cout<<"Positon out of range"<<endl;
 }
}

void single_llist::delete_pos()
{
   int pos, i, counter = 0;
 if (start == NULL)
 {
 cout<<"List is empty"<<endl;
 return;
 }

cout<<"Enter the position of value to be deleted: ";
 cin>>pos;
 struct node *s, *ptr;
 s = start;
 if (pos == 1)
 {
 start = s->next;
```

```
}
else
{
while (s != NULL)
{
s = s->next;
counter++;
}
if (pos > 0 && pos <= counter)
{
s = start;
for (i = 1;i < pos;i++)
{
ptr = s;
s = s->next;
}
ptr->next = s->next;
}
else
{
cout<<"Position out of range"<<endl;
}
```

```
 free(s);
 cout<<"Element Deleted"<<endl;
 }
}

void single_llist::search()
{
 int value, pos = 0;
 bool flag = false;
 if (start == NULL)

 {
 cout<<"List is empty"<<endl;
 return;
 }
 cout<<"Enter the value to be searched: ";
 cin>>value;
 struct node *s;
 s = start;
 while (s != NULL)
 {
 pos++;
```

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

void single_llist::display()
{
 struct node *temp;
 if (start == NULL)
 {
 cout<<"The List is Empty"<<endl;
 return;
 }
 temp = start;
 cout<<"Elements of list are: "<<endl;
 while (temp != NULL)
```

```
{
cout<<temp->info<<"->";
temp = temp->next;
}
cout<<"NULL"<<endl;
}
```

OUTPUT:

```
------------------------
Operations on singly linked list
------------------------
1.Insert node at position
2.Delete an existing element
3.locations of a given element
4.Display Linked List
5.Exit
Enter your choice:1
Inserting Node at a given position:
Enter the value to be inserted: 5
Enter the postion at which node to be inserted: 1

------------------------
Operations on singly linked list
------------------------
1.Insert node at position
2.Delete an existing element
3.locations of a given element
4.Display Linked List
5.Exit
Enter your choice:1
Inserting Node at a given position:
Enter the value to be inserted: 6
Enter the postion at which node to be inserted: 2
Positon out of range

------------------------
Operations on singly linked list
------------------------
1.Insert node at position
2.Delete an existing element
3.locations of a given element
4.Display Linked List
5.Exit
```

```
Enter your choice:1
Inserting Node at a given position:
Enter the value to be inserted: 7
Enter the postion at which node to be inserted: 3
Positon out of range

------------------------
Operations on singly linked list
------------------------
1.Insert node at position
2.Delete an existing element
3.locations of a given element
4.Display Linked List
5.Exit
Enter your choice:2
Delete a particular node:
Enter the position of value to be deleted: 3
Position out of range
Element Deleted
------------------------
Operations on singly linked list
------------------------
1.Insert node at position
2.Delete an existing element
3.locations of a given element
4.Display Linked List
5.Exit
Enter your choice:3
Search element in Link List:
Enter the value to be searched: 5
Element 5 is found at position1
```