PRACTICAL-7

Write a menu driven program to implement following operations on the singly linked list.

(a) Insert a node at the front of the linked list.

(b) Insert a node at the end of the linked list.

(c) Insert a node such that linked list is in ascending order.(according to info. Field)

(d) Delete a first node of the linked list.

(e) Delete a node before specified position.

(f) Delete a node after specified position.

SOURCE CODE:

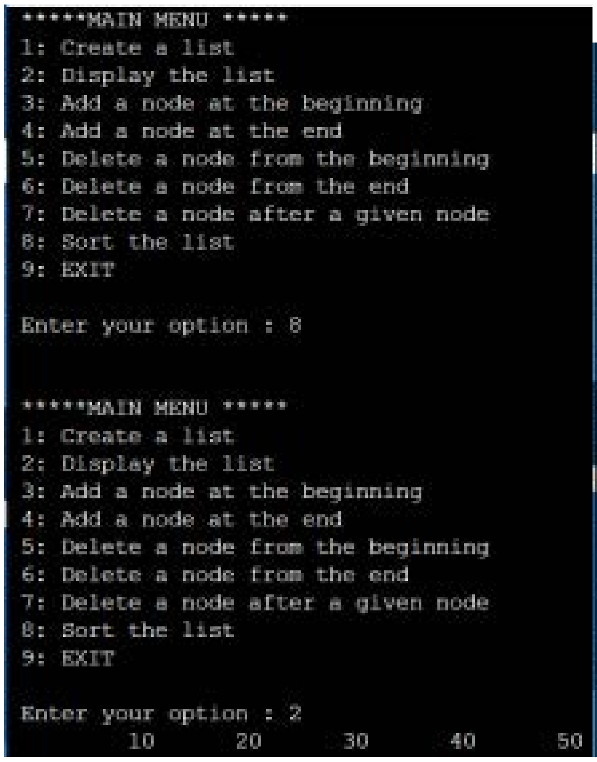
```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
struct node
{
int data;
struct node *next;
};
struct node *start = NULL;
struct node *create_ll(struct node *);
struct node *display(struct node *);
struct node *insert_beg(struct node *);
struct node *insert_end(struct node *);
struct node *delete_beg(struct node *);
struct node *delete_end(struct node *);
struct node *delete_after(struct node *);
struct node *sort_list(struct node *);
int main(int argc, char *argv[])
{
int option;
do
{
printf("\n\n *****MAIN MENU *****");
printf("\n 1: Create a list");
printf("\n 2: Display the list");
printf("\n 3: Add a node at the beginning");
printf("\n 4: Add a node at the end");
printf("\n 5: Delete a node from the beginning");
printf("\n 6: Delete a node from the end");
printf("\n 7: Delete a node after a given node");

printf("\n 8: Sort the list");
printf("\n 9: EXIT");
printf("\n\n Enter your option : ");
scanf("%d", &option);
switch(option)
{
case 1: start = create_ll(start);
printf("\n LINKED LIST CREATED");
break;
case 2: start = display(start);
break;
case 3: start = insert_beg(start);
break;
case 4: start = insert_end(start);
break;
case 5: start = delete_beg(start);
break;
case 6: start = delete_end(start);
break;
case 7: start = delete_after(start);
break;
case 8: start = sort_list(start);
break;
}
}
while(option !=13);
return 0;
}
struct node *create_ll(struct node *start)
{
struct node *new_node, *ptr;
int num;
printf("\n Enter -1 to end");
printf("\n Enter the data : ");
scanf("%d", &num);
```

```
while(num!=-1)

{

new_node = (struct node*)malloc(sizeof(struct node));

new_node -> data=num;

if(start==NULL)

{

new_node -> next = NULL;

start = new_node;

}

else

{

ptr=start;

while(ptr->next!=NULL)

ptr=ptr->next;

ptr->next = new_node;

new_node->next=NULL;


}

printf("\n Enter the data : ");

scanf("%d", &num);

}

return start;

}

struct node *display(struct node *start)

{

struct node *ptr;

ptr = start;
```

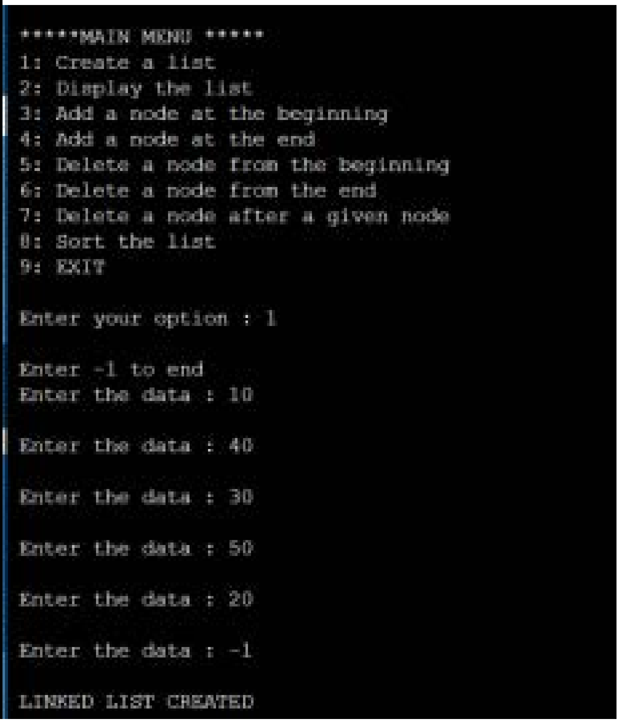
```
while(ptr != NULL)
{
printf("\t %d", ptr -> data);
ptr = ptr -> next;
}
return start;
}
struct node *insert_beg(struct node *start)
{
struct node *new_node;
int num;
printf("\n Enter the data : ");
scanf("%d", &num);
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> data = num;
new_node -> next = start;
start = new_node;
return start;
}
struct node *insert_end(struct node *start)
{
struct node *ptr, *new_node;
int num;
printf("\n Enter the data : ");
scanf("%d", &num);
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> data = num;
new_node -> next = NULL;
ptr = start;
while(ptr -> next != NULL)
ptr = ptr -> next;
ptr -> next = new_node;
return start;
}
struct node *delete_beg(struct node *start)
{
struct node *ptr;
ptr = start;

start = start -> next;
free(ptr);
return start;
}
struct node *delete_end(struct node *start)
{
struct node *ptr, *preptr;
ptr = start;
while(ptr -> next != NULL)
{
preptr = ptr;
ptr = ptr -> next;
}
preptr -> next = NULL;
free(ptr);
```

```
return start;

}

struct node *delete_after(struct node *start)

{

struct node *ptr, *preptr;

int val;

printf("\n Enter the value after which the node has to deleted : ");

scanf("%d", &val);

ptr = start;

preptr = ptr;

while(preptr -> data != val)

{

preptr = ptr;

ptr = ptr -> next;

}

preptr -> next=ptr -> next;

free(ptr);

return start;

}

struct node *sort_list(struct node *start)

{

struct node *ptr1, *ptr2;

int temp;

ptr1 = start;

while(ptr1 -> next != NULL)

{

ptr2 = ptr1 -> next;
while(ptr2 != NULL)

{

if(ptr1 -> data > ptr2 -> data)

{

temp = ptr1 -> data;

ptr1 -> data = ptr2 -> data;

ptr2 -> data = temp;


}

ptr2 = ptr2 -> next;

}

ptr1 = ptr1 -> next;

}

return start;

}
```

OUTPUT:



```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Delete a node from the beginning
6: Delete a node from the end
7: Delete a node after a given node
8: Sort the list
9: EXIT

Enter your option : 8


*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Delete a node from the beginning
6: Delete a node from the end
7: Delete a node after a given node
8: Sort the list
9: EXIT

Enter your option : 2
        10      20      30      40      50
```



```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Delete a node from the beginning
6: Delete a node from the end
7: Delete a node after a given node
8: Sort the list
9: EXIT

Enter your option : 1

Enter -1 to end
Enter the data : 10

Enter the data : 40

Enter the data : 30

Enter the data : 50

Enter the data : 20

Enter the data : -1

LINKED LIST CREATED
```