SOURCE CODE: SINGLY LINKED LIST:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<stdlib.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 *insert_bef(struct node *);
struct node *insert_aft(struct node *);
struct node *delete_beg(struct node *);
struct node *delete_end(struct node *);
struct node *delete_node(struct node *);
struct node *delete_aft(struct node *);
struct node *delete_list(struct node *);
struct node *sort_list(struct node *);
int main()
{
        int option;
        clrscr();
        do
        {
```

```
		printf("\n**************Main Menu************\n");
		printf("1.Create a list\n2.Display the list\n3.Add a node at the beginning\n4.Add the node at the end\n5.Add a node before a given node\n6.Add a node after a given node\n7.Delete a node from beginning\n8.Delete a node from the end\n9.Delete a given node\n10.Delete a node after a given node\n11.Delete a entire list\n12.Sort the list\n13.Exit\n\nEnter your option\n");
		scanf("%d",&option);
		switch(option)
		{
			case 1:
				start=create_ll(start);
				printf("Linked list created\n");
				break;
			case 2:
				start=display(start);
				break;
			case 3:
				start=insert_beg(start);
				break;
			case 4:
				start=insert_end(start);
				break;
			case 5:
				start=insert_bef(start);
				break;
			case 6:
				start=insert_aft(start);
				break;
			case 7:
				start=delete_beg(start);
				break;
```

```
                        case 8:
                                start=delete_end(start);
                                break;
                        case 9:
                                start=delete_node(start);
                                break;
                        case 10:
                                start=delete_aft(start);
                                break;
                        case 11:
                                start=delete_list(start);
                                printf("Linked list delted\n");
                                break;
                        case 12:
                                start=sort_list(start);
                                break;
                }
        }while(option!=13);
        getch();
        return 0;
}
struct node *create_ll(struct node *start)
{
        struct node *new_node,*ptr;
        int num;
        printf("Enter the data\n");
        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("Enter the data\n");
		scanf("%d",&num);
	}
	return start;
}
struct node *display(struct node *start)
{
	struct node *ptr;
	ptr=start;
	while(ptr!=NULL)
	{
		printf("%d\t",ptr->data);
		ptr=ptr->next;
	}
```

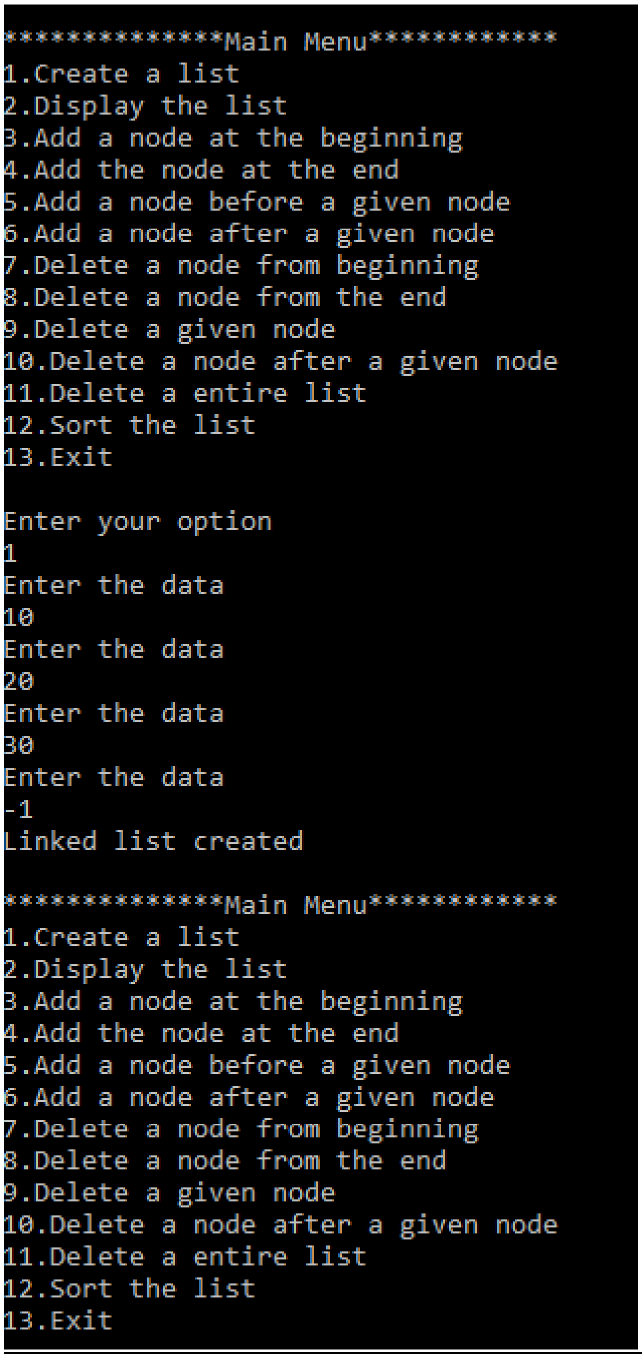
```
        return start;
}
struct node *insert_beg(struct node *start)
{
        struct node *new_node;
        int num;
        printf("Enter the data\n");
        scanf("%d",&num);
        new_node=(struct node *)malloc(sizeof(struct node));
        new_node->data=num;
        new_node->next=start;
        start=new_node;
        return start;
}
struct node *insert_end(struct node *start)
{
        struct node *new_node,*ptr;
        int num;
        printf("Enter the data\n");
        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 *insert_bef(struct node *start)
{
        struct node *new_node,*ptr,*preptr;
        int num,val;
        printf("Enter the data\n");
        scanf("%d",&num);
        printf("Enter the value before which data has to be added\n");
        scanf("%d",&val);
        new_node=(struct node *)malloc(sizeof(struct node));
        new_node->data=num;
        ptr=start;
        while(ptr->data!=val)
        {
                preptr=ptr;
                ptr=ptr->next;
        }
        preptr->next=new_node;
        new_node->next=ptr;
        return start;
}
struct node *insert_aft(struct node *start)
{
        struct node *new_node,*ptr,*postptr;
        int num,val;
        printf("Enter the data\n");
        scanf("%d",&num);
        printf("Enter the value after which data has to be added\n");
        scanf("%d",&val);
        new_node=(struct node *)malloc(sizeof(struct node));
        new_node->data=num;
        ptr=start;
        postptr=ptr;
        while(postptr->data!=val)
        {
                postptr=ptr;
                ptr=ptr->next;
        }
        postptr->next=new_node;
        new_node->next=ptr;
        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_node(struct node *start)
{
        struct node *ptr,*preptr;
        int val;
        printf("Enter the value of the node which has to be deleted\n");
        scanf("%d",&val);
        ptr=start;
        if(ptr->data==val)
        {
                start=delete_beg(start);
                return start;
        }
        else
        {
                while(ptr->data!=val)
                {
                        preptr=ptr;
                        ptr=ptr->next;
                }
                preptr->next=ptr->next;
                free(ptr);
                return start;
        }
}
struct node *delete_aft(struct node *start)
```

```
{
	struct node *ptr,*preptr;
	int val;
	printf("Enter the value of after which the node has to be deleted\n");
	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 *delete_list(struct node *start)
{
	struct node *ptr;
	if(start!=NULL)
	{
		ptr=start;
		while(ptr!=NULL)
		{
			printf("%d is to be deleted\n",ptr->data);
			start=delete_beg(ptr);
			ptr=start;
		}
	}
```

```
        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;
}
```

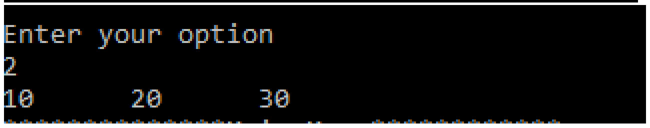
OUTPUT:



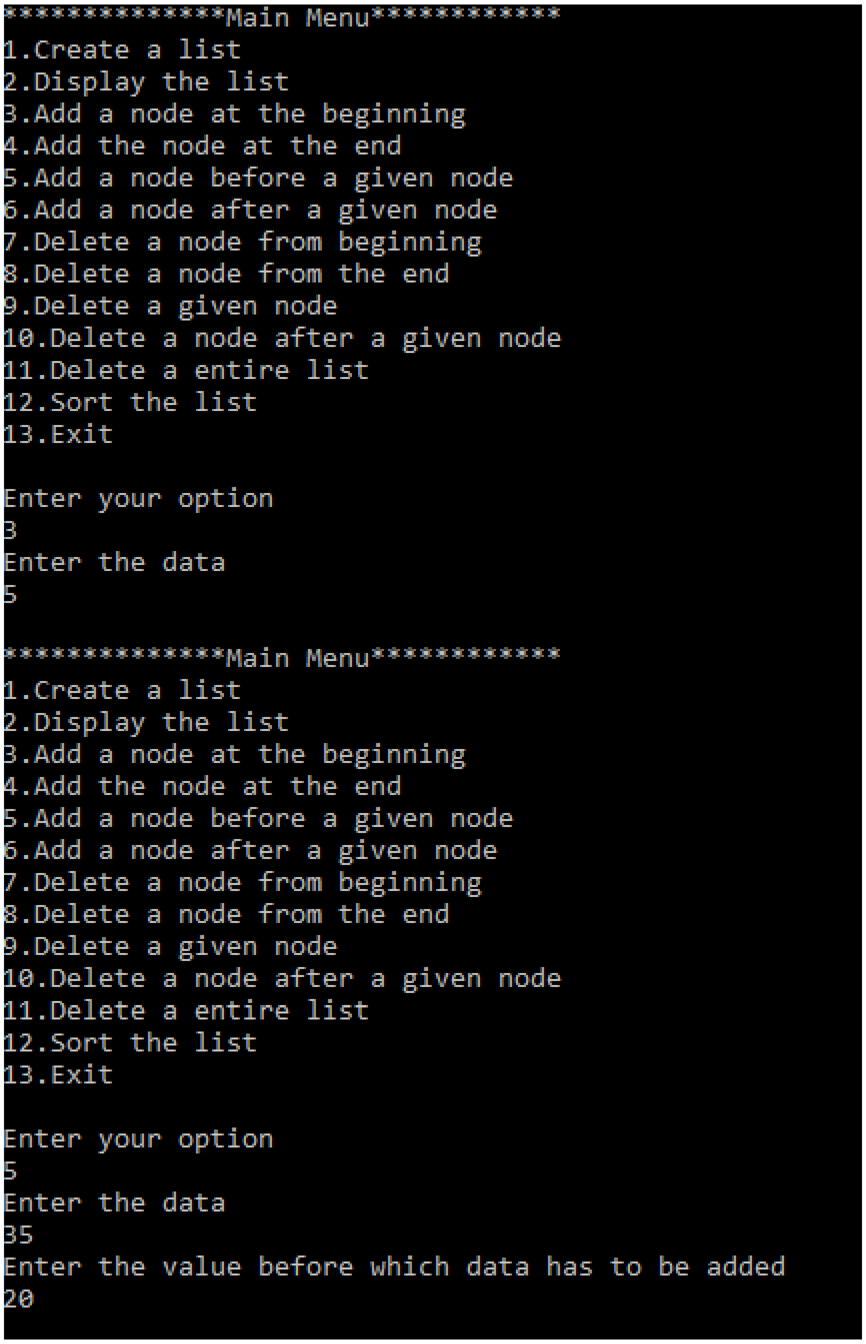
```
**************Main Menu*************
1.Create a list
2.Display the list
3.Add a node at the beginning
4.Add the node at the end
5.Add a node before a given node
6.Add a node after a given node
7.Delete a node from beginning
8.Delete a node from the end
9.Delete a given node
10.Delete a node after a given node
11.Delete a entire list
12.Sort the list
13.Exit

Enter your option
1
Enter the data
10
Enter the data
20
Enter the data
30
Enter the data
-1
Linked list created

**************Main Menu*************
1.Create a list
2.Display the list
3.Add a node at the beginning
4.Add the node at the end
5.Add a node before a given node
6.Add a node after a given node
7.Delete a node from beginning
8.Delete a node from the end
9.Delete a given node
10.Delete a node after a given node
11.Delete a entire list
12.Sort the list
13.Exit
```



```
Enter your option
2
10      20      30
```

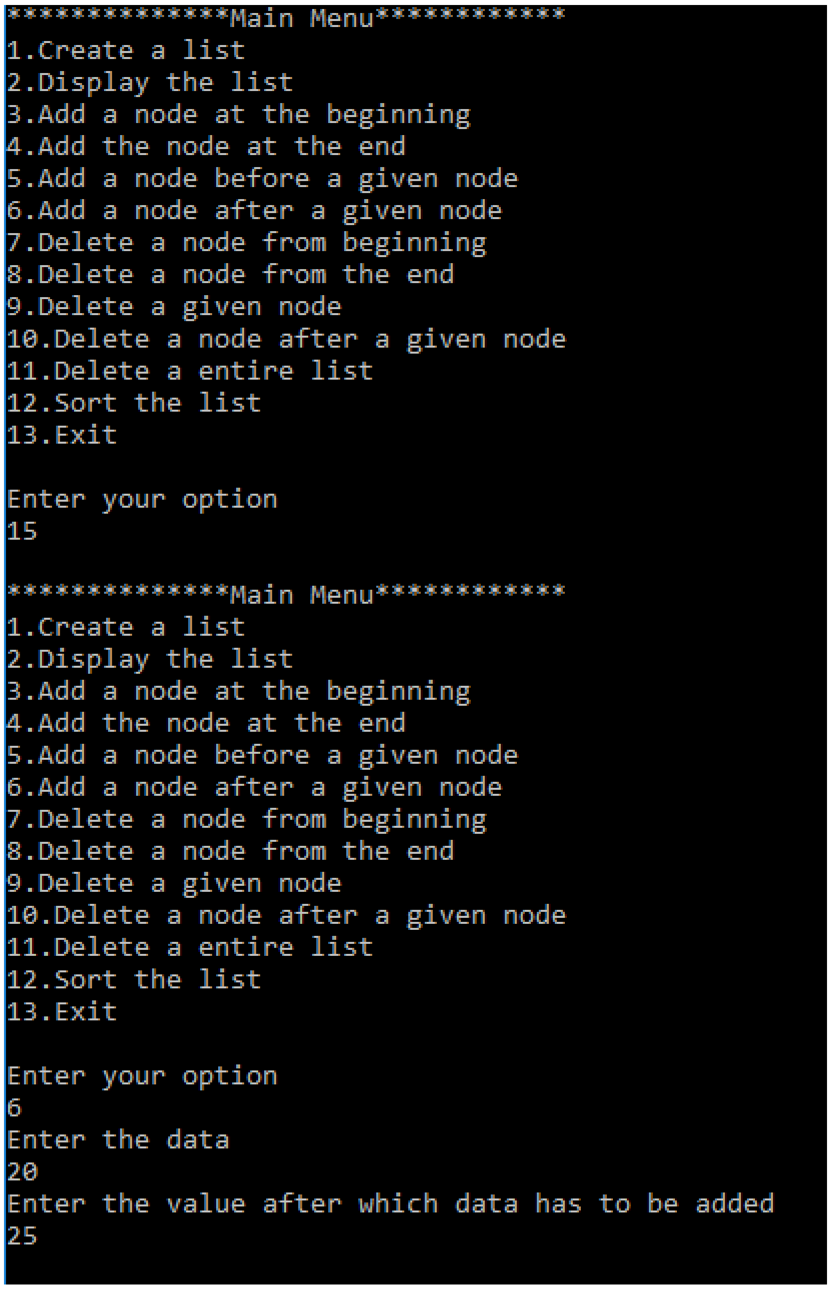


```
**************Main Menu************
1.Create a list
2.Display the list
3.Add a node at the beginning
4.Add the node at the end
5.Add a node before a given node
6.Add a node after a given node
7.Delete a node from beginning
8.Delete a node from the end
9.Delete a given node
10.Delete a node after a given node
11.Delete a entire list
12.Sort the list
13.Exit

Enter your option
3
Enter the data
5

**************Main Menu************
1.Create a list
2.Display the list
3.Add a node at the beginning
4.Add the node at the end
5.Add a node before a given node
6.Add a node after a given node
7.Delete a node from beginning
8.Delete a node from the end
9.Delete a given node
10.Delete a node after a given node
11.Delete a entire list
12.Sort the list
13.Exit

Enter your option
5
Enter the data
35
Enter the value before which data has to be added
20
```


```
**************Main Menu************
1.Create a list
2.Display the list
3.Add a node at the beginning
4.Add the node at the end
5.Add a node before a given node
6.Add a node after a given node
7.Delete a node from beginning
8.Delete a node from the end
9.Delete a given node
10.Delete a node after a given node
11.Delete a entire list
12.Sort the list
13.Exit

Enter your option
15

**************Main Menu************
1.Create a list
2.Display the list
3.Add a node at the beginning
4.Add the node at the end
5.Add a node before a given node
6.Add a node after a given node
7.Delete a node from beginning
8.Delete a node from the end
9.Delete a given node
10.Delete a node after a given node
11.Delete a entire list
12.Sort the list
13.Exit

Enter your option
6
Enter the data
20
Enter the value after which data has to be added
25
```