

**/\*linked list full op\*/**

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
struct node
```

```
{
```

```
    int data;
```

```
    struct node*link;
```

```
};
```

```
struct node*header;
```

```
struct node*create_sll(struct node*);
```

```
struct node*display(struct node*);
```

```
struct node*insert_beg(struct node*);
```

```
struct node*insert_end(struct node*);
```

```
struct node*insert_any(struct node*);
```

```
struct node*delete_beg(struct node*);
```

```
struct node*delete_end(struct node*);
```

```
struct node*delete_any(struct node*);
```

```
void search();
```

```
struct node*sort_list(struct node*);
```

```
int main()
```

```
{
```

```
    int ch;
```

```
    while(ch!=11)
```

```
    {
```

```
        printf("MAIN MENU\n");
```

```
        printf("1.create the list\n2.display the list\n3.insert at beg\n4.insert at end\n5.insert  
at any position\n6.delete at beg\n7.delete at end\n8.delete from any position\n9.search\n10.sort  
the list\n11.exit\n");
```

```
        printf("enter your choice\n");
```

```
        scanf("%d",&ch);
```

```
        switch(ch)
```

```

        {

            case 1:header=create_sll(header);
            break;

            case 2:header=display(header);
            break;

            case 3:header=insert_beg(header);
            break;

            case 4:header=insert_end(header);
            break;

            case 5:header=insert_any(header);
            break;

            case 6:header=delete_beg(header);
            break;

            case 7:header=delete_end(header);
            break;

            case 8:header=delete_any(header);
            break;

            case 9:search();
            break;

            case 10:header=sort_list(header);
            break;

            case 11:exit(0);

            default:

                printf("invalid choice\n");

        }

    }

}

struct node*create_sll(struct node*header)
{

    struct node*new_node,*ptr;

    int item;

```

```

printf("enter -1 to end\n");

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

scanf("%d",&item);

while(item!=-1)
{
    new_node=(struct node*)malloc(sizeof(struct node));

    new_node->data=item;

    if(header==NULL) //list is empty
    {
        new_node->link=NULL;

        header=new_node;
    }
    else
    {
        ptr=header;

        while(ptr->link!=NULL)
        {
            ptr=ptr->link;
        }

        ptr->link=new_node;

        new_node->link=NULL;
    }

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

    scanf("%d",&item);
}

printf("list created\n");

return header;
}

struct node*display(struct node*header)
{
    printf("the list is below\n");

```

```

    struct node*ptr;

    ptr=header;

    while(ptr!=NULL)
    {
        printf("%d\n",ptr->data);
        ptr=ptr->link;
    }

    return header;
}

struct node*insert_beg(struct node*header)
{
    struct node*new_node;
    int item;

    if(header==NULL)    //memory bank returns null
    {
        printf("over flow ,insertion not possible\n");
    }
    else
    {
        printf("enter the data to be inserted: \n");
        scanf("%d",&item);
        new_node=(struct node*)malloc(sizeof(struct node));
        new_node->data=item;
        new_node->link=header;
        header=new_node;
        printf("node imserted at the beg\n");
    }

    return header;
}

struct node*insert_end(struct node*header)
{

```

```

struct node*new_node,*ptr;

int item;

if(header==NULL)    //memory bank returns null
{
    printf("over flow ,insertion not possible\n");
}
else
{
    printf("enter the data to be inserted: \n");
    scanf("%d",&item);
    new_node=(struct node*)malloc(sizeof(struct node));
    new_node->data=item;
    ptr=header;
    while(ptr->link!=NULL)
    {
        ptr=ptr->link;
    }
    ptr->link=new_node;
    new_node->link=NULL;
    printf("node inserted at the end\n");
}
return header;
}

struct node*insert_any(struct node*header)
{
    struct node*new_node,*ptr;
    int item,loc,i;
    if(header==NULL)    //memory bank returns null
    {
        printf("over flow ,insertion not possible\n");
    }

```

```

else
{
    printf("enter the location at which the data has to be inserted: \n");
    scanf("%d",&loc);
    printf("enter the data to be inserted: \n");
    scanf("%d",&item);
    new_node=(struct node*)malloc(sizeof(struct node));
    new_node->data=item;
    ptr=header;
    for(i=0;i<loc-1;i++)
    {
        ptr=ptr->link;
    }
    new_node->link=ptr->link;
    ptr->link=new_node;
    printf("the node is inserted at specific position\n");
}
return header;
}

struct node*delete_beg(struct node*header)
{
    struct node*ptr;
    if(header==NULL) //list is empty
    {
        printf("deletion not possible\n");
    }
    else
    {
        ptr=header;
        header=header->link;
        free(ptr);
    }
}

```

```

        printf("node deleted from the beg\n");
    }
    return header;
}

struct node*delete_end(struct node*header)
{
    struct node*ptr,*ptr1;
    if(header==NULL) //list is empty
    {
        printf("deletion not possible\n");
    }
    else
    {
        ptr=header;
        while(ptr->link!=NULL)
        {
            ptr1=ptr;
            ptr=ptr->link;
        }
        ptr1->link=NULL;
        free(ptr);
        printf("node deleted from end\n");
    }
    return header;
}

struct node*delete_any(struct node*header)
{
    struct node*ptr,*ptr1;
    int loc,i;
    if(header==NULL) //list is empty
    {

```

```

        printf("deletion not possible\n");
    }
    else
    {
        printf("enter the location after which the node has to be deleted: \n");
        scanf("%d",&loc);
        ptr=header;
        for(i=0;i<=loc;i++)
        {
            ptr1=ptr;
            ptr=ptr->link;
        }
        ptr1->link=ptr->link;
        free(ptr);
        printf("the node is deleted from the specific position\n");
    }
    return header;
}

void search()
{
    struct node*ptr;
    int item,flag=0,loc,i=0;
    if(header==NULL)
    {
        printf("list is empty\n");
    }
    else
    {
        printf("enter the data to be searched: \n");
        scanf("%d",&item);
        ptr=header;

```



```

        while(ptr!=NULL)
        {
            if(ptr->data==item)
            {
                flag=1;
                loc=i+1;
                break;
            }
            else
            {
                flag=0;
            }
            ++i;
            ptr=ptr->link;
        }
        if(flag==0)
        {
            printf("search item not found\n");
        }
        else
        {
            printf("item found at location:%d\n",loc);
        }
    }
}

struct node*sort_list(struct node*header)
{
    struct node*ptr1,*ptr2;
    int temp;
    ptr1=header;
    while(ptr1->link!=NULL)

```

```
{  
    ptr2=ptr1->link;  
    while(ptr2!=NULL)  
    {  
        if(ptr1->data>ptr2->data)  
        {  
            temp=ptr1->data;  
            ptr1->data=ptr2->data;  
            ptr2->data=temp;  
        }  
        ptr2=ptr2->link;  
    }  
    ptr1=ptr1->link;  
}  
printf("the list is sorted\n");  
return header;  
}
```

```
C:\Users\HP\OneDrive\Desktop\some rest ll prog\ll full op with insertion at position.exe
MAIN MENU
1.create the list
2.display the list
3.insert at beg
4.insert at end
5.insert at any position
6.delete at beg
7.delete at end
8.delete from any position
9.search
10.sort the list
11.exit
enter your choice
1
enter -1 to end
enter the data:
10
enter the data:
20
enter the data:
30
enter the data:
-1
list created
MAIN MENU
1.create the list
2.display the list
3.insert at beg
4.insert at end
5.insert at any position
6.delete at beg
7.delete at end
8.delete from any position
9.search
10.sort the list
11.exit
enter your choice
2
the list is below
10
20
30
MAIN MENU
1.create the list
2.display the list
3.insert at beg
4.insert at end
5.insert at any position
6.delete at beg
7.delete at end
8.delete from any position
9.search
10.sort the list
11.exit
enter your choice
5
enter the location at which the data has to be inserted:
1
enter the data to be inserted:
15
the node is inserted at specific position
MAIN MENU
1.create the list
2.display the list
3.insert at beg
4.insert at end
5.insert at any position
6.delete at beg
7.delete at end
8.delete from any position
9.search
10.sort the list
11.exit
enter your choice
2
the list is below
10
15
20
30
MAIN MENU
1.create the list
2.display the list
3.insert at beg
4.insert at end
5.insert at any position
6.delete at beg
7.delete at end
8.delete from any position
9.search
10.sort the list
11.exit
enter your choice
11
```

```
C:\Users\HP\OneDrive\Desktop\some rest ll prog\ll full op with insertion at position.exe
enter your choice
3
enter the location at which the data has to be inserted:
1
enter the data to be inserted:
15
the node is inserted at specific position
MAIN MENU
1.create the list
2.display the list
3.insert at beg
4.insert at end
5.insert at any position
6.delete at beg
7.delete at end
8.delete from any position
9.search
10.sort the list
11.exit
enter your choice
2
the list is below
10
15
20
00
MAIN MENU
1.create the list
2.display the list
3.insert at beg
4.insert at end
5.insert at any position
6.delete at beg
7.delete at end
8.delete from any position
9.search
10.sort the list
11.exit
enter your choice
11
-----
Process exited after 23.08 seconds with return value 0
Press any key to continue . . .
```