Program to create a CLL and perform insertion and deletion at beginning and end

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *link;
};
struct node *root=NULL;
struct node *current;
void create();
void append();
int display(struct node*);
void insert_begin();
int length();
void delete_begin();
void delete_end();
void add_after();
void delete_specified();
int main()
{
    int op,len;
    printf("**menu**\n");
    printf("1.create\n");
    printf("2.display\n");
    printf("3.append\n");
    printf("4.insert_begin\n");
    printf("5.length\n");
    printf("6.delete_begin\n");
    printf("7.delete_end\n");
    printf("8.add_after\n");
    printf("9.delete_specified\n");
    printf("10.exit\n");
    while(1)
    {
        printf("\nenter option\n");
        scanf("%d",&op);
        switch(op)
        {
            case 1:
                    create();
                    break;
            case 2:
                    display(root);
```

```
                break;
            case 3:
                append();
                break;
            case 4:
                insert_begin();
                break;
            case 5:
                len=length();
                printf("length:%d nodes\n",len);
                break;
            case 6:
                delete_begin();
                break;
            case 7:
                delete_end();
                break;
            case 8:
                add_after();
                break;
            case 9:
                delete_specified();
                break;
            case 10:
                exit(1);
                break;


        }
    }

}
void create()
{
    struct node *temp,*current;
    int num;
    printf("enter -1 to exit\n");
    printf("enter data\n");
    scanf("%d",&num);
    while(num!=-1)
    {
        temp=(struct node*)malloc(sizeof(struct node));
        temp->data=num;
        if(root==NULL)
        {
            root=temp;
            root->link=root;
            current=root;
```

```
        }
        else
        {
            while(current->link!=root)
            {
                current=current->link;
            }
            current->link=temp;
            temp->link=root;
        }
        printf("enter data\n");
        scanf("%d",&num);
    }

}
int display(struct node *temp)
{
    printf("elements in list\n");
    while(temp->link!=root)
    {
        printf("%d\t",temp->data);
        temp=temp->link;
    }
    printf("%d",temp->data);
}
void append()
{
    struct node *temp,*current;
    temp=(struct node*)malloc(sizeof(struct node));
    printf("enter new data\n");
    scanf("%d",&temp->data);
    current=root;
    while(current->link!=root)
    {
        current=current->link;
    }
    current->link=temp;
    temp->link=root;


}
void insert_begin()
{
    struct node *temp,*current=root;
    temp=(struct node*)malloc(sizeof(struct node));
    printf("enter new data\n");
    scanf("%d",&temp->data);
    while(current->link!=root)
```

```
    {
        current=current->link;
    }
    temp->link=root;
    current->link=temp;
    root=temp;
}
int length()
{
    struct node *current=root;
    int c=0;
    if(root==NULL)
    {
        c=-1;
    }
    else
    {

        while(current->link!=root)
        {
            c++;
            current=current->link;
        }
    }
    return c+1;

}
void delete_begin()
{
    struct node *temp=root;
    struct node *current=root;
    while(current->link!=root)
    {
        current=current->link;
    }
    current->link=temp->link;
    root=temp->link;
    temp->link=NULL;
}
void delete_end()
{
    struct node *current,*current_bef;
    current=root;
    while(current->link!=root)
    {
        current_bef=current;
        current=current->link;
    }
```

```
    current_bef->link=current->link;
    free(current);
}
void add_after()
{
    int loc,i=1;

    printf("enter location\n");
    scanf("%d",&loc);
    if(loc>length())
    {
        printf("invalid location\n");
    }
    else
    {
        struct node *p=root,*temp;
        temp=(struct node*)malloc(sizeof(struct node));
        printf("enter new data\n");
        scanf("%d",&temp->data);
        temp->link=NULL;
        while(i<loc)
        {
            p=p->link;
            i++;
        }
        temp->link=p->link;
        p->link=temp;


    }
}
void delete_specified()
{
    int loc,i=1;
    printf("enter location\n");
    scanf("%d",&loc);
    if(loc>length())
    {
        printf("invalid location\n");
    }
    else
    {
        struct node *p,*q;
        p=root;
        while(i<loc-1)
        {
            p=p->link;
            i++;
```

```
        }
        q=p->link;
        p->link=q->link;
        q->link=NULL;
        free(q);

    }
}
```

Output:

```
**menu**
1.create
2.display
3.append
4.insert_begin
5.length
6.delete_begin
7.delete_end
8.add_after
9.delete_specified
10.exit

enter option
1
enter -1 to exit
enter data
5
enter data
6
enter data
7
enter data
9
enter data
-1
enter option
2
elements in list
5       6       7       9
enter option
3
enter new data
0

enter option   //display
2
```

```
elements in list
5       6       7       9       0
enter option    //exit
10
```