PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int inf;
struct node *link;
}*header,*p,*p1,*temp;
Void insert_beg();
Void insert_end();
Void insert_any();
Void delete_beg();
Void delete_end();
Void display();
Void main()
{
int ch;
clrscr();
```

```
while(1)
{
printf("\n1. insert at beginning");
printf("\n2. insert at end");
printf("\n3. insert at any");
printf("\n4. delete at beg");
printf("\n5. delete. at end");
printf("\n6. delete. at any");
printf("\n7. display");
printf("\n8. exit");
printf("\n enter your choice\n");
 scanf("%d", &ch);
switch(ch)
 {
case 1: insert_beg();
        break;
case 2: insert_end();
        break;
case 3: insert_any();
        break;
case 4: delete_beg();
        break;
case 5: delete_end();
        break;
case 6: delete_any();
```

```
            break;
case 7: display();
            break;
case 8: exit(0);
default : printf("\n invalid choice");
            break;
}
}
}
Void insert_beg()
{
  temp=(struct node*)malloc(sizeof(struct node));
  printf("\n enter element");
  scanf("%d",&temp->inf);
  if(temp==NULL)
  {
  printf("\n insertion is not possible");
  return;
  }
  else
  {
  temp->link=header;
  header=temp;
 // temp->inf=NULL;
  }
```

```
}
Void insert_end()
{
temp=(struct node *) malloc(sizeof(struct node));
printf("\n enter the element");
scanf("%d", &temp->info);
if(temp==NULL)
  {
  printf("\n insertion is not possible");
  return;
  }
else
 {
 p= header;
  while(p->link!=NULL)
  {
  p=p->link;
  }
  p->link=temp;
  temp->link=NULL;
}
 }
Void insert_any()
{
int key;
```

```
temp=(struct node.*)malloc(size.of(struct node));
printf("\n enter the element");
scanf("%d", &temp->inf);
printf("\n enter the key");
scanf("%d", &key);
if(temp==NULL)
 {
 printf("\n insertion is not possible.");
 return;
 }
else
{
p=header;
while((p->inf!=key)&&(p->link!=NULL))
 {
 p=p->link;
 }
 if((p->link==NULL)&&(p->inf!=key))
 {
 printf("\n key is not available in the list");
 return;
 }
 else
 {
 temp->link=p->link;
```

```
//temp->inf = NULL;
p->link = temp;
 }
 }
}
Void delete_beg()
{
p=header;
 if(p==NULL)
  {
printf("In the list is empty");
return;
 }
 else
 {
 p1 = p->link;
 header = p1;
  free(p);
  }
 }
Void delete_end()
{
 p=header;
  if(p==NULL)
   {
```

```
printf("\n the list is empty");
return;
}
else
{
p1 = p -> link;
header = p1;
free(p);
}
}
Void delete_end()
{
p = header;
if (p == NULL)
{
printf("\n the list is empty");
return;
}
else
{
while(p -> link != NULL)
{
p1 = p;
p = p -> link;
}
```

```
p1 -> link = NULL;
free(p);
 }
}
Void delete_any()
{
 int key;
 p = header;
 printf("\n enter the key");
 scanf("%d", &key);
 while(p != NULL)
  {
 if(p -> info != key)
  {
 p1 = p;
  p = p -> link;
   }
 else
   {
 if(header -> info == key)
 {
 header = p -> link;
 }
 else
 p1 -> link = p -> link;
```

```
free(p);
return;
}
}
if(p!=NULL)
{
printf("\n node with %d does not exist", key);
}
}
Void display()
{
p=header;
if(p!=NULL)
{
printf("\n\nADDRESS\t DATA\t LINK\n\n");
while(p!=NULL)
{
printf("\n 0x%x\t %d\t 0x%x\n", p, p->inf,
                                  p->link);
p=p->link;
}}
else
printf("\n list is empty"); }
```