CIRCULAR SINGLE LINKED(DATA,NAME,MARKS) LIST AND OPERATIONS

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   char name[20];
   int marks;
   struct node *next;
};
struct node *head=NULL;
void create()
{
 struct node *second,*third,*fourth,*fifth;
 clrscr();
 head=malloc(sizeof(struct node));
 second=malloc(sizeof(struct node));
 third=malloc(sizeof(struct node));
 fourth=malloc(sizeof(struct node));
 fifth=malloc(sizeof(struct node));
 head->data=10;
 strcpy(head->name,"siva");
 head->marks=100;
 second->data=20;
 strcpy(second->name,"pawan");
 second->marks=200;
 third->data=30;
 strcpy(third->name,"gopi");
 third->marks=300;
 fourth->data=40;
```

```
 strcpy(fourth->name,"sudha");
 fourth->marks=400;
 fifth->data=50;
 strcpy(fifth->name,"murari");
 fifth->marks=500;
 head->next=second;
 second->next=third;
 third->next=fourth;
fourth->next=fifth;
fifth->next=head;
}
void addFirst(int val,char name[],int marks)
{
struct node *newnode=malloc(sizeof(struct node));
newnode->data=val;
strcpy(newnode->name,name);
newnode->marks=marks;
if(head==NULL)
{
newnode->next=newnode;
head=newnode;
}
else
{
struct node *temp;
temp=head;
while(temp->next!=head)
{
temp=temp->next;
}
```

```
temp->next=newnode;
newnode->next=head;
head=newnode;
}
}
void addEnd(int val,char name[],int marks)
{
struct node *newnode=malloc(sizeof(struct node));
newnode->data=val;
strcpy(newnode->name,name);
newnode->marks=marks;
if(head==NULL)
{
newnode->next=newnode;
head=newnode;
}
else
{
struct node *temp;
temp=head;
while(temp->next!=head)
{
temp=temp->next;
}
temp->next=newnode;
newnode->next=head;
}
}
void addposition(int val,char name[],int marks)
{
```

```
int pos,i=1;
struct node  *newnode,*temp;
newnode=malloc(sizeof(struct node));
newnode->data=val;
strcpy(newnode->name,name);
newnode->marks=marks;
newnode->next=NULL;
printf("enter the position");
scanf("%d",&pos);
temp=head;
while(i<pos)
{
temp=temp->next;
i++;
}
newnode->next=temp->next;
temp->next=newnode;
}
void deleteBEP(int val)
{
struct node *current,*temp;
temp=head;
if(temp->data==val)
{
current=head;
while(current->next!=head)
{
current=current->next;
}
head=head->next;
```

```
current->next=head;
free(temp);
}
else
{
current=head;
while(current->next!=head)
{
if(current->next->data==val)
{
temp=current->next;
current->next=temp->next;
free(temp);
break;
}
else
{
current=current->next;
}
}
}
}
int search(int val)
{
struct node *temp;
temp=head;
while(temp!=head)
{
if(temp->data==val)
   return 1;
```

```
temp=temp->next;
}
return -1;
}
void display()
{
struct node *temp;
temp=head;
do
{
printf("%d%s%d--",temp->data,temp->name,temp->marks);
temp=temp->next;
}while(temp!=head);
}
void main()
{
int op,val,marks,x;
char name[20];
   create();
   while(1)
   {

         printf("\n linked list menu\n");
         printf("1.list is created plz enter dispaly option\n");
         printf("2.insert at begin\n");
         printf("3.insert at end\n");
         printf("4.insert at pos\n");
         printf("5.delete at beg,end,pos\n");
         printf("6.search\n");
         printf("7.display\n");
```

```
printf("8.exit\n");
printf("choose one option\n");
scanf("%d",&op);
switch(op)
{
case 1:printf("\n list is created plz enter display option");
case 2:printf("\n enter value name marks");
      scanf("%d%s%d",&val,name,&marks);
      addFirst(val,name,marks);
      break;
case 3:printf("\n enter a value name marks");
      scanf("%d%s%d",&val,name,&marks);
      addEnd(val,name,marks);
      break;
case 4:printf("\n enter a value name marks");
      scanf("%d%s%d",&val,name,&marks);
      addposition(val,name,marks);
      break;
case 5:printf("\n enter a value");
      scanf("%d",&val);
      deleteBEP(val);
      break;
case 6:printf("\n enter a value");
      scanf("%d",&val);
      x=search(val);
      if(x==1)
      printf("\n element found");
      else
      printf("\n element not found");
      break;
```

```
        case 7:
              display();
              break;
        case 8:
              exit(1);
              break;
        default:
           printf("choose valid option\n");

        }

   }
}
```