

//Singly Circular Linked List

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

struct student
{
    int rollno;
    char name[10];
    struct student *next;
};

struct student *start;

void main()
{
    int choice;
    void insert_first();
    void insert_last();
    void insert_specific();
    void delete_first();
    void delete_last();
    void delete_specific_value();
    void delete_specific_nodeno();
    void display();
    void search();
    void sort();

    do
    {
        clrscr();

        printf("\n\t1. Insert First");
        printf("\n\t2. Insert Last");
        printf("\n\t3. Insert Specific");
        printf("\n\t4. Delete First");
        printf("\n\t5. Delete Last");
        printf("\n\t6. Delete Specific by Value");
        printf("\n\t7. Delete Specific by Node No");
        printf("\n\t8. Display");
        printf("\n\t9. Search");
        printf("\n\t10. Sort");
        printf("\n\t0. Exit");

        printf("\n\tEnter your choice : ");
        scanf("%d",&choice);

        switch(choice)
        {
            case 1:
                insert_first();
                break;
            case 2:
                insert_last();
                break;
```

```

case 3:
    insert_specific();
    break;
case 4:
    delete_first();
    break;
case 5:
    delete_last();
    break;
case 6:
    delete_specific_value();
    break;
case 7:
    delete_specific_nodeno();
    break;
case 8:
    display();
    break;
case 9:
    search();
    break;
case 10:
    sort();
    break;
case 0:
    printf("\n\tEnd of the program");
    break;
default:
    printf("\n\tInvalid Choice");
    break;
}
getch();
}
while(choice != 0);
}

```

```

void insert_first()
{
    struct student *newnode, *temp;

    if(start == NULL)
    {
        start = (struct student *) malloc(sizeof(struct student));

        printf("\n\tEnter Roll No. : ");
        scanf("%d",&start->rollno);

        printf("\n\tEnter Name : ");
        fflush(stdin);
        gets(start->name);

        start->next = start;
    }
    else
    {

```

```
newnode = (struct student *) malloc(sizeof(struct student));
```

```
printf("\n\tEnter Roll No. : ");  
scanf("%d",&newnode->rollno);
```

```
printf("\n\tEnter Name : ");  
fflush(stdin);  
gets(newnode->name);
```

```
temp = start;  
while(temp->next != start)  
{  
    temp = temp->next;  
}
```

```
newnode->next = start;  
start = newnode;  
temp->next = start;  
}  
}
```

```
void insert_last()
```

```
{  
    struct student *newnode;  
  
    if(start == NULL)  
    {  
        start = (struct student *) malloc(sizeof(struct student));  
        newnode = start;  
        start->next = start;  
    }  
    else  
    {  
        newnode = start;  
        while(newnode->next != start)  
        {  
            newnode = newnode->next;  
        }  
    }
```

```
newnode->next = (struct student *) malloc(sizeof(struct student));  
newnode = newnode->next;  
newnode->next = start;  
}
```

```
printf("\n\tEnter Roll No. : ");  
scanf("%d",&newnode->rollno);
```

```
printf("\n\tEnter Name : ");  
fflush(stdin);  
gets(newnode->name);  
}
```

```
void insert_specific()
```

```
{  
    int a, count=1, nodeno;
```

```
struct student *temp, *newnode;
```

```
if(start == NULL)
```

```
{  
    insert_first();  
}
```

```
else
```

```
{  
    temp = start;  
    while(temp->next != start)  
    {  
        temp = temp->next;  
        count++;  
    }  
  
    do
```

```
{
```

```
    printf("\n\tEnter Node no. to Insert between 1 to %d : ", count+1);  
    scanf("%d",&nodeno);  
}
```

```
while(nodeno < 1 || nodeno > count+1);
```

```
if(nodeno == 1)
```

```
{  
    insert_first();  
}
```

```
else if(nodeno == count + 1)
```

```
{  
    insert_last();  
}
```

```
else
```

```
{  
    temp = start;  
    a = 1;
```

```
while(a < nodeno - 1)
```

```
{  
    temp = temp->next;  
    a++;  
}
```

```
newnode = (struct student *) malloc(sizeof(struct student));
```

```
newnode->next = temp->next;
```

```
temp->next = newnode;
```

```
printf("\n\tEnter Roll No. : ");
```

```
scanf("%d",&newnode->rollno);
```

```
printf("\n\tEnter Name : ");
```

```
fflush(stdin);
```

```
gets(newnode->name);
```

```
}
```

```
}
```

```
}
```

```

void delete_first()
{
    struct student *deletenode, *temp;

    if(start == NULL)
    {
        printf("\n\tSingly Circular Linked List is Empty");
    }
    else
    {
        deletenode = start;
        if(start->next == start)
        {
            start = NULL;
        }
        else
        {
            temp = start;
            while(temp->next != start)
            {
                temp = temp->next;
            }
            temp->next = start->next;

            start = start->next;
        }
        printf("\n\tDelete Node Information = ");
        printf("\n\tRoll No. = %d",deletenode->rollno);
        printf("\n\tName = %s",deletenode->name);

        free(deletenode);
    }
}

```

```

void delete_last()
{
    struct student *temp, *deletenode;
    if(start == NULL)
    {
        printf("\n\tSingly Circular Linked List is Empty");
    }
    else
    {
        if(start->next == start)
        {
            deletenode = start;
            start = NULL;
        }
        else
        {
            temp = start;

            while(temp->next->next != start)
            {
                temp = temp->next;
            }

```

```

}

deletenode = temp->next;
temp->next = temp->next->next;
}

printf("\n\tDelete Node Information = ");
printf("\n\tRoll No. = %d",deletenode->rollno);
printf("\n\tName = %s",deletenode->name);

free(deletenode);
}
}

void delete_specific_value()
{
int no,flag=0;
struct student *temp, *deletenode;

if(start == NULL)
{
printf("\n\tSingly Circular Linked List is Empty");
}
else
{
printf("\n\tEnter Roll No. to Delete : ");
scanf("%d",&no);

temp = start;
if(start->rollno == no)
{
delete_first();
}
else
{
temp = start;
while(temp->next != start)
{
if(temp->next->rollno == no)
{
deletenode = temp->next;
temp->next = temp->next->next;

printf("\n\tDelete Node Information = ");
printf("\n\tRoll No. = %d",deletenode->rollno);
printf("\n\tName = %s",deletenode->name);

flag = 1;
free(deletenode);
break;
}
temp = temp->next;
}

if(flag == 0)

```

```

    {
        printf("\n\tRoll No. %d not found", no);
    }
}
}
}
}

```

```

void delete_specific_nodeno()

```

```

{
    int a, nodeno, count = 1;
    struct student *temp, *deletenode;

```

```

    if(start == NULL)
    {
        printf("\n\tSingly Linked List is Empty");
    }

```

```

    else
    {
        temp = start;

```

```

        while(temp->next != start)
        {
            count++;
            temp = temp->next;
        }

```

```

        do
        {
            printf("\n\tEnter node no to delete between 1 to %d : ",count);
            scanf("%d",&nodeno);
        }
        while(nodeno < 1 || nodeno > count);

```

```

        if(nodeno == 1)
        {
            delete_first();
        }
        else if(nodeno == count)
        {
            delete_last();
        }
        else
        {
            temp = start;
            a = 1;
            while(a < nodeno-1)
            {
                temp = temp->next;
                a++;
            }

```

```

            deletenode = temp->next;
            temp->next = temp->next->next;

```

```

            printf("\n\tDelete Node Information = ");

```

```

printf("\n\tRoll No. = %d",deletenode->rollno);
printf("\n\tName = %s",deletenode->name);

free(deletenode);
}
}
}

void display()
{
    struct student *temp;

    if(start == NULL)
    {
        printf("\n\tSingly Circular Linked List is Empty");
    }
    else
    {
        temp = start;

        printf("\n\tRoll No.\tName\n");

        while(temp->next != start)
        {
            printf("\n\t%d\t\t%s",temp->rollno, temp->name);
            temp = temp->next;
        }
        printf("\n\t%d\t\t%s",temp->rollno, temp->name); //Print's Last Node information
    }
}

void search()
{
    struct student *temp=start;
    int no,flag = 0;

    if(start == NULL)
    {
        printf("\n\tSingly Circular Linked List is Empty");
    }
    else
    {
        printf("\n\tEnter Roll No. to Search : ");
        scanf("%d",&no);

        if(start->rollno == no)
        {
            printf("\n\tRecord Found");
            printf("\n\tRoll No. = %d",start->rollno);
            printf("\n\tName = %s",start->name);
            flag = 1;
        }
        else
        {
            temp = start;

```



```

while(temp->next != start)
{
    if(temp->next->rollno == no)
    {
        printf("\n\tRecord Found");
        printf("\n\tRoll No. = %d",temp->next->rollno);
        printf("\n\tName = %s",temp->next->name);
        flag = 1;
        break;
    }
    temp = temp->next;
}
}
if(flag == 0)
{
    printf("\n\tRoll No. %d not found", no);
}
}
}

```

```

void sort()
{
    struct student *temp1,*temp2;
    int no;
    char nm[10];

    if(start == NULL)
    {
        printf("\n\tSingly Circular Linked List is Empty");
    }
    else
    {
        temp1 = start;

        while(temp1->next != start)
        {
            temp2 = temp1->next;
            while(temp2 != start)
            {
                if(temp1->rollno > temp2->rollno)
                {
                    no = temp1->rollno;
                    temp1->rollno = temp2->rollno;
                    temp2->rollno = no;

                    strcpy(nm, temp1->name);
                    strcpy(temp1->name, temp2->name);
                    strcpy(temp2->name, nm);
                }
                temp2 = temp2->next;
            }
            temp1 = temp1->next;
        }
        display();
    }
}

```

}
}