circular singly linked list

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node *next;

};

struct node *head;



void beginsert ();

void lastinsert ();

void randominsert();

void begin_delete();

void last_delete();

void random_delete();

void display();

void search();

void main ()

{

    int choice =0;

    while(choice != 9)

    {
        printf("\n*********Main Menu*********\n");

        printf("\nChoose one option from the following list ...\n");

        printf("\n===============================================\n");

        printf("\n1.Insert in begining\n2.Insert at last\n3.random insrtion\n4.Delete from Beginning\n5.Delete from 
last\n6.random deletion\n7.Search for an element\n8.Show\n9.Exit\n");

        printf("\nEnter your choice?\n");

        scanf("\n%d",&choice);

        switch(choice)

        {

          case 1:

          beginsert();

          break;

          case 2:

          lastinsert();

          break;

                              case 3:

                              randominsert();

                              break;

          case 4:

          begin_delete();

          break;

          case 5:

          last_delete();

          break;
```

```
                    case 6:

                    random_delete();

                    break;

          case 7:

          search();

          break;

          case 8:

          display();

          break;

          case 9:

          exit(0);

          break;

          default:

          printf("Please enter valid choice..");

        }

    }

}

void beginsert()

{

    struct node *ptr,*temp;

    int item;

    ptr = (struct node *)malloc(sizeof(struct node));

    if(ptr == NULL)

    {
```

```
    printf("\nOVERFLOW");
}
else
{
    printf("\nEnter the node data?");
    scanf("%d",&item);
    ptr -> data = item;
    if(head == NULL)
    {
        head = ptr;
        ptr -> next = head;
    }
    else
    {
        temp = head;
        while(temp->next != head)
            temp = temp->next;
        ptr->next = head;
        temp -> next = ptr;
        head = ptr;
    }
    printf("\nnode inserted\n");
}
```

```
}

void lastinsert()

{

    struct node *ptr,*temp;

    int item;

    ptr = (struct node *)malloc(sizeof(struct node));

    if(ptr == NULL)

    {

        printf("\nOVERFLOW\n");

    }

    else

    {

        printf("\nEnter Data?");

        scanf("%d",&item);

        ptr->data = item;

        if(head == NULL)

        {

            head = ptr;

            ptr -> next = head;

        }

        else

        {

            temp = head;

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

        printf("\nnode inserted\n");
    }

}
 void randominsert()
{
int item,loc,i;
struct node *ptr,*temp;
ptr=(struct node*)malloc(sizeof(struct node*));
if(ptr==NULL)
             {
             printf("\nover flow");
             }
else
             {
             printf("enter a number to be inserted:");
             scanf("%d",&item);
```

```
          ptr->data=item;

          printf("enter location where node has to be inserted:\n");

          scanf("%d",&loc);

          temp=head;

          for(i=1;i<loc;i++)

          {

          temp=temp->next;

           if(temp==head)

             {

                      printf("can't inserted\n");

                      return;

                       }

    }

    ptr->next=temp->next;

    temp->next=ptr;

    printf(" node inserted\n");

}

}

void begin_delete()

{

    struct node *ptr;

    if(head == NULL)

    {

        printf("\nUNDERFLOW");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }

    else
    {   ptr = head;
        while(ptr -> next != head)
            ptr = ptr -> next;
        ptr->next = head->next;
        free(head);
        head = ptr->next;
        printf("\nnode deleted\n");

    }
}
void last_delete()
{
    struct node *ptr, *preptr;
    if(head==NULL)
    {
        printf("\nUNDERFLOW");
    }
    else if (head ->next == head)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");

    }
    else
    {
        ptr = head;
        while(ptr ->next != head)
        {
            preptr=ptr;
            ptr = ptr->next;
        }
        preptr->next = ptr -> next;
        free(ptr);
        printf("\nnode deleted\n");

    }
}
  void random_delete()
```

```
{
            struct node *ptr,*ptr1;
            int loc,i;
            printf("Enter the location of node when you want to perform deletion:");
            scanf("%d",&loc);
            ptr=head;
            for(i=1;i<loc;i++)
            {
                        ptr1=ptr;
                        ptr=ptr->next;
            if(ptr==head)
            {
                        printf("can't delete\n");
                        return;
            }
    }
    ptr1->next=ptr->next;
    free(ptr);
    printf("deleted node is %d\n",loc);
}
void search()
{
    struct node *ptr;
            int i=0,item;
```

```
ptr = head;

if(ptr == NULL)

{

    printf("\nEmpty List\n");

}

else

{

    printf("\nEnter item which you want to search?\n");

    scanf("%d",&item);

    if(head ->data == item)

    {

    printf("item found at location %d",1);

           }

    else

    {

    while (ptr->next != head)

    {

          i++;

       if(ptr->data == item)

        {

            printf("item found at location %d ",i);

            break;

        }
```

```
            ptr = ptr -> next;

        }

        if(ptr->data==item&&ptr->next==head)

        {

             printf("item found at location %d ",i+1);

                          }

        else if(ptr->data!=item)

        {

            printf("Item not found\n");

        }

    }

}



}



void display()

{

    struct node *ptr;

    ptr=head;

    if(head == NULL)

    {

        printf("\nnothing to print");

    }

    else
    {

        printf("\n printing values ... \n");



        while(ptr -> next != head)

        {



            printf("%d\n", ptr -> data);

            ptr = ptr -> next;

        }

        printf("%d\n", ptr -> data);

    }



}
```