CIRCULAR DOUBLY LINKED LIST

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

    struct node *prev;

    struct node *next;

    int data;

};

struct node *head;

void insertion_beginning();

void insertion_last();

void random_insertion();

void deletion_beginning();

void deletion_last();

void random_deletion();

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 Beginning\n2.Insert at last\n3.random insertion\n4.Delete from
Beginning\n5.Delete from last\n6.random deletion\n7.Search\n8.Show\n9.Exit\n");

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

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

        switch(choice)
```

```
{
    case 1:
    insertion_beginning();
    break;
    case 2:
            insertion_last();
    break;
                    case 3:
                    random_insertion();
                    break;
    case 4:
    deletion_beginning();
    break;
    case 5:
    deletion_last();
    break;
                    case 6:
                    random_deletion();
                    break;
    case 7:
    search();
    break;
    case 8:
    display();
    break;
    case 9:
    exit(0);
    break;
    default:
    printf("Please enter valid choice..");
}
```

```
    }
}
void insertion_beginning()
{
   struct node *ptr,*temp;
   int item;
   ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW");
   }
   else
   {
    printf("\nEnter Item value");
    scanf("%d",&item);
    ptr->data=item;
   if(head==NULL)
   {
      head = ptr;
      ptr -> next = head;
      ptr -> prev = head;
   }
   else
   {
       temp = head;
    while(temp -> next != head)
    {
        temp = temp -> next;
    }
    temp -> next = ptr;
    ptr -> prev = temp;
```

```
      head -> prev = ptr;
      ptr -> next = head;
      head = ptr;
     }
     printf("\nNode inserted\n");
 }


 }
 void insertion_last()
 {
     struct node *ptr,*temp;
     int item;
     ptr = (struct node *) malloc(sizeof(struct node));
     if(ptr == NULL)
     {
         printf("\nOVERFLOW");
     }
     else
     {
         printf("\nEnter value");
         scanf("%d",&item);
          ptr->data=item;
         if(head == NULL)
         {
             head = ptr;
             ptr -> next = head;
             ptr -> prev = head;
         }
         else
         {
            temp = head;
            while(temp->next !=head)
            {
                temp = temp->next;
            }
            temp->next = ptr;
            ptr ->prev=temp;
            head -> prev = ptr;
        ptr -> next = head;
          }
     }
       printf("\nnode inserted\n");
 }
   void random_insertion()
   {
            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;
     ptr->prev=temp;
     temp->next->prev=ptr;
     temp->next=ptr;
     printf(" node inserted\n");
 }
 }
 void deletion_beginning()
 {
     struct node *temp,*ptr;
     if(head == NULL)
     {
         printf("\n UNDERFLOW");
     }
     else if(head->next == head)
     {
         head = NULL;
         free(head);
         printf("\nnode deleted\n");
     }
     else
     {
         temp = head;
                     ptr=head ;
```

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

}
void deletion_last()
{
    struct node *ptr,*temp;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }
    else
    {
        ptr = head;
        while (ptr->next != head)
        {
            temp = ptr;
            ptr = ptr->next;
        }
        temp->next = head;
        head->prev = temp;
        free(ptr);
        printf("\nnode deleted\n");

     }
  }
  void random_deletion()
  {
          struct node *ptr,*temp;
          int var;
          printf("Enter the value of var:");
          scanf("%d",&var);
          ptr=head;
          while(ptr->data!=var)
          {
                  ptr=ptr->next;
          if(ptr==head)
          {
                  printf("can't delete\n");
                  return;
          }
     }
     if(ptr->next==head)
    {
          ptr->prev->next=head;
          head->prev=ptr->prev;
     free(ptr);
     printf("deleted element is %d\n",var);
    }
```

```
   else
   {
          temp=ptr->next;
          ptr->prev->next=temp;
          temp->prev=ptr->prev;
          free(ptr);
          printf("deleted node is %d\n",var);
   }
  }
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);
    }

}
```

```
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");
          }
      }
}

}
```