Aim:- Implementation of Circular Singly Linked List.

Code:-

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node *head;

void beg_insert ();
void last_insert ();
void randominsert();
void begin_delete();
void last_delete();
void random_delete();
void display();
void search();
void main ()
{   printf("D10A_Atharva Chavan_9\n");
printf("**** Implementation of Circular Singly Linked List****\n");
    int choice =0;
```

```
    while(choice != 7)
    {   printf("\n1. Insert in begining\n2. Insert at last\n3. Delete from
Beginning\n4. Delete from last\n5. Search for an element\n6. Show\n7.
Exit\n");
        printf("\nEnter your choice:");
        scanf("\n%d",&choice);
        switch(choice)
        {
            case 1:
            beg_insert();
            break;
            case 2:
            last_insert();
            break;
            case 3:
            begin_delete();
            break;
            case 4:
            last_delete();
            break;
            case 5:
            search();
            break;
            case 6:
            display();
```

```
            break;
            case 7: printf("You have exited the linked list!\n");
            exit(0);
            break;
            default:
            printf("Please enter valid choice..");
        }
    }
}
void beg_insert()
{
    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 last_insert()
{
    struct node *ptr,*temp;
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW\n");
    }
```

```
    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;
            }
            temp -> next = ptr;
            ptr -> next = head;
        }

        printf("\nnode 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 search()
{
    struct node *ptr;
    int item,i=0,flag=1;
    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",i+1);
        flag=0;
        }
        else
        {
```

```
        while (ptr->next != head)
        {
            if(ptr->data == item)
            {
                printf("item found at location %d ",i+1);
                flag=0;
                break;
            }
            else
            {
                flag=1;
            }
            i++;
            ptr = ptr -> next;
        }
        }
        if(flag != 0)
        {
            printf("Item not found\n");
        }
    }

}

void display()
```

```
{
    struct node *ptr;
    ptr=head;
    if(head == NULL)
    {
        printf("\nnothing to print");
    }
    else
    {
        printf("\nThe elements inside present are: \n");

        while(ptr -> next != head)
        {

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

}
```

Output:-

```
/tmp/y1ueV93Y93.o
D10A_Atharva Chavan_9
**** Implementation of Circular Singly Linked List****

1. Insert in begining
2. Insert at last
3. Delete from Beginning
4. Delete from last
5. Search for an element
6. Show
7. Exit

Enter your choice:1
Enter the node data:10
Node inserted

1. Insert in begining
2. Insert at last
3. Delete from Beginning
4. Delete from last
5. Search for an element
6. Show
7. Exit

Enter your choice:1
Enter the node data:20
Node inserted
```

```
1. Insert in begining
2. Insert at last
3. Delete from Beginning
4. Delete from last
5. Search for an element
6. Show
7. Exit

Enter your choice:2
Enter the Node Data:30
node inserted

1. Insert in begining
2. Insert at last
3. Delete from Beginning
4. Delete from last
5. Search for an element
6. Show
7. Exit

Enter your choice:4
node deleted

1. Insert in begining
2. Insert at last
3. Delete from Beginning
4. Delete from last
```

```
5. Search for an element
6. Show
7. Exit

Enter your choice:6
The elements inside present are:
20
10

1. Insert in begining
2. Insert at last
3. Delete from Beginning
4. Delete from last
5. Search for an element
6. Show
7. Exit

Enter your choice:7
You have exited the linked list!
```