Aim:- Implementation of Circular Doubly Linked List.

Code:-

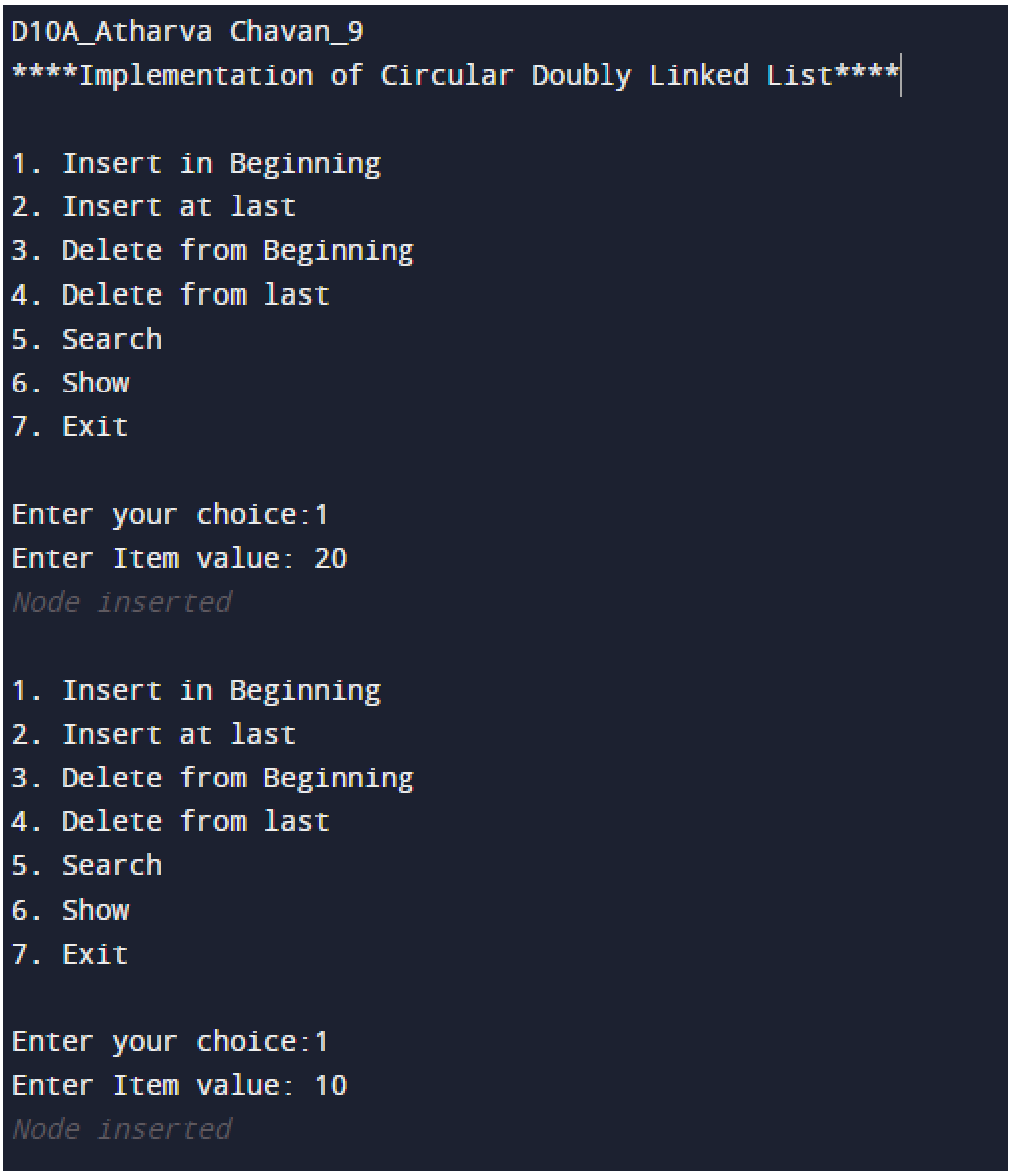
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    struct node *prev;
    struct node *next;
    int data;
};
struct node *head;
void insertion_beginning();
void insertion_last();
void deletion_beginning();
void deletion_last();
void display();
void search();
void main ()
{  printf("D10A_Atharva Chavan_9\n");
printf("****Implementation of Circular Doubly Linked List****\n");
int choice =0;
    while(choice != 9)
    {   printf("\n1. Insert in Beginning\n2. Insert at last\n3. Delete from
Beginning\n4. Delete from last\n5. Search\n6. Show\n7. Exit\n");

        printf("\nEnter your choice:");
        scanf("\n%d",&choice);
        switch(choice)
        {
            case 1:
            insertion_beginning();
            break;
            case 2:
                    insertion_last();
            break;
            case 3:
            deletion_beginning();
            break;
            case 4:
            deletion_last();
            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 insertion_beginning()
{
   struct node *ptr,*temp;
   int item;
   ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW\n");
   }
   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;
```

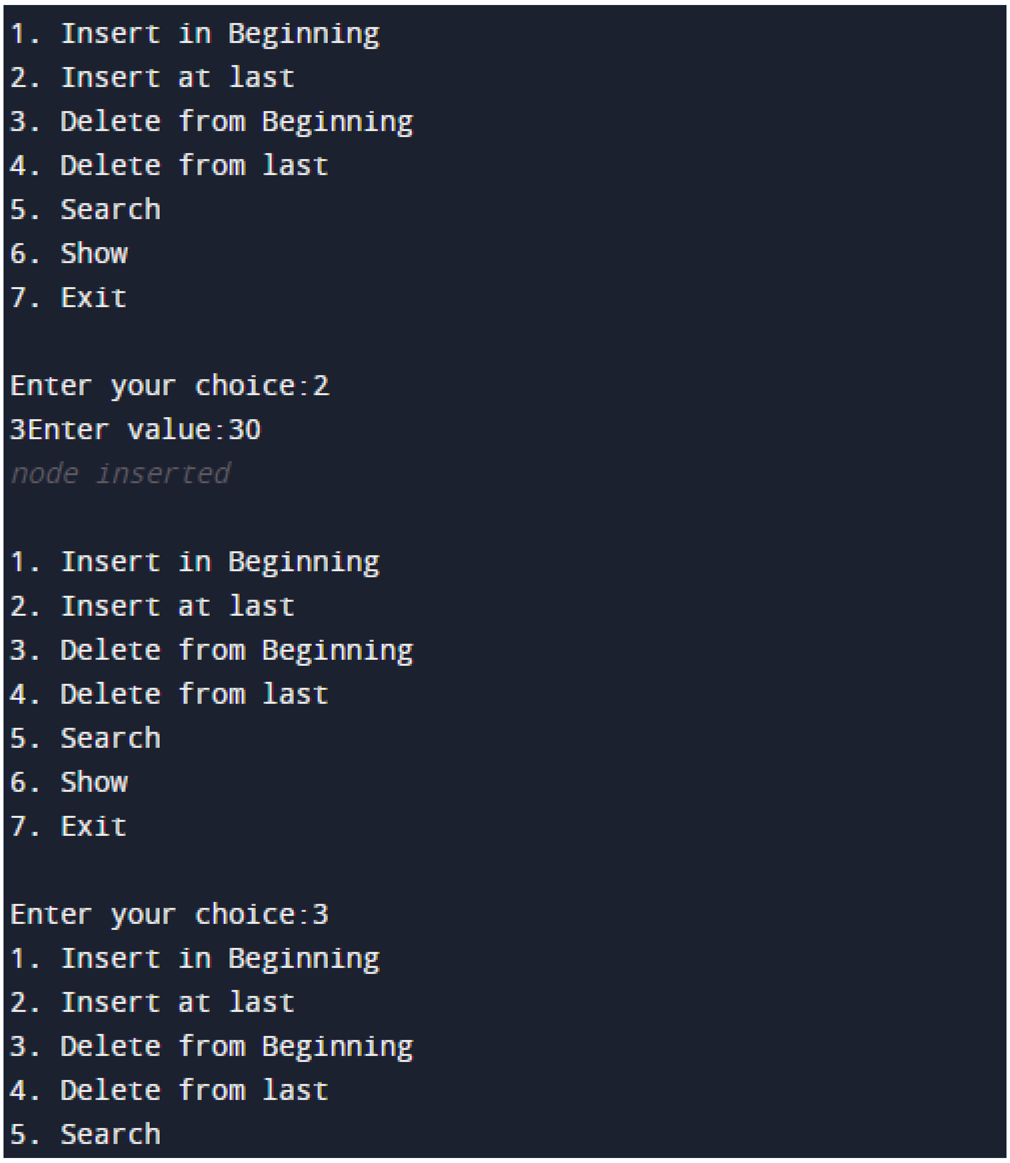
```
      ptr -> next = head;
        }
   }
     printf("\nnode inserted\n");
}

void deletion_beginning()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("\nUNDERFLOW");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nNode deleted\n");
    }
    else
    {
        temp = head;
        while(temp -> next != head)
        {
            temp = temp -> next;
        }
        temp -> next = head -> next;
        head -> next -> prev = temp;
        free(head);
        head = temp -> next;
    }

}
void deletion_last()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\nUNDERFLOW");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nNode deleted\n");
    }
    else
    {
        ptr = head;
        if(ptr->next != head)
        {
            ptr = ptr -> next;
        }
        ptr -> prev -> next = head;
        head -> prev = ptr -> prev;
        free(ptr);
        printf("\nNode deleted\n");
    }
}

void display()
{
    struct node *ptr;
    ptr=head;
    if(head == NULL)
    {
        printf("\nNothing to print");
    }
    else
    {
        printf("\nThe elements present here: \n");

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

}

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");
        }
    }
```

```
}
```

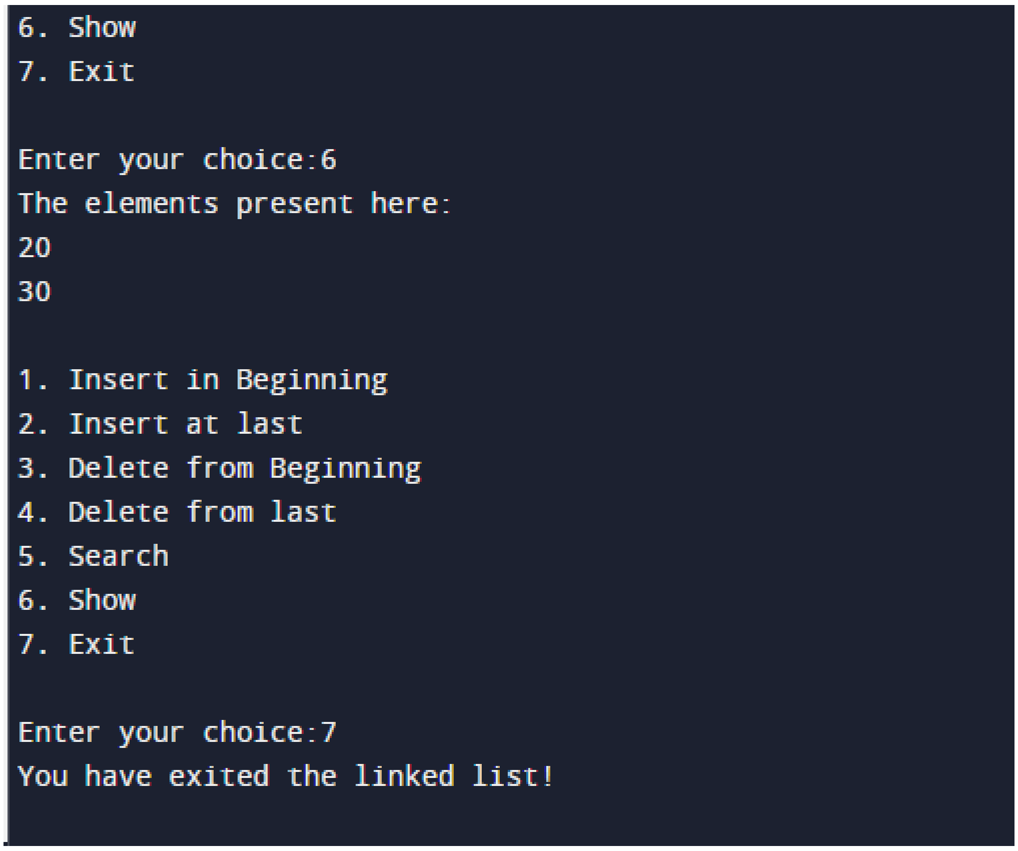
Output:-



```
D10A_Atharva Chavan_9
****Implementation of Circular Doubly Linked List****

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

Enter your choice:1
Enter Item value: 20
Node inserted

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

Enter your choice:1
Enter Item value: 10
Node inserted
```



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

Enter your choice:2
3Enter value:30
node inserted

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

Enter your choice:3
1. Insert in Beginning
2. Insert at last
3. Delete from Beginning
4. Delete from last
5. Search
```



```
6. Show
7. Exit

Enter your choice:6
The elements present here:
20
30

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

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