```
// Created by Ritwik Chandra Pandey on 18/02/21.
// 183215
// Double Circular Linked List Implementation

#include<stdio.h>
#include<stdlib.h>

struct node {
    int data;
    struct node *prev;
    struct node *next;
};
typedef struct node *NODE;
NODE first = NULL;

NODE createNodeInDCLL(){
    NODE temp;
    temp=(NODE)malloc(sizeof(struct node));
    temp->prev=NULL;
    temp->next=NULL;
    return temp;
}

NODE addNodesInDCLL(NODE first, int x) {
    NODE temp, lastNode = first;
    temp = createNodeInDCLL();
    temp -> data = x;
    if (first == NULL) {
        first = temp;
    } else {
        lastNode = first -> prev;
        lastNode -> next = temp;
        temp -> prev = lastNode;
    }
    temp -> next = first;
    first -> prev = temp;
```

```
    return first;
}

void traverseListInDCLL(NODE first) {
    NODE temp = first;
    do {
        printf("%d <--> ", temp -> data);
        temp = temp -> next;
    } while (temp != first);
    printf("END");
    printf("\n");
}



int searchPosOfEleInDCLL(NODE first, int key) {
    NODE currentNode = first;
    int count = 0;
    if (currentNode == NULL) {
        return count;
    }
    while(currentNode != NULL && currentNode -> data != key) {
        if(currentNode -> next == first) {
            return 0;
        }
        count++;
        currentNode = currentNode -> next;
    }
    return(count + 1);
}

NODE insertAtBeginInDCLL(NODE first, int x) {
    NODE temp, lastNode;
    temp = createNodeInDCLL();
    temp -> data = x;
    if (first == NULL) {
        temp -> next = temp;
```

```
        temp -> prev = temp;
    } else {
        lastNode = first-> prev;
        temp -> prev = lastNode;
        temp -> next = first;
        lastNode -> next = temp;
        first -> prev = temp;
    }
    first = temp;
    return first;
}
NODE insertAtEndInDCLL(NODE first, int x) {
    NODE temp, lastNode = first;
    temp = createNodeInDCLL();
    temp -> data = x;
    if (first == NULL) {
        first = temp;
    } else {
        lastNode = first -> prev;
        lastNode -> next = temp;
        temp -> prev = lastNode;
    }
    temp -> next = first;
    first -> prev = temp;
    return first;
}

NODE insertAtPositionInDCLL(NODE first, int pos, int x) {
    NODE temp, lastNode = first;
    int i;
    for (i = 1; i < (pos - 1); i++) {
        if (lastNode -> next == first) {
            printf("No such position in Doubly Circular Linked List so insertion is not possible\n");
            return first;
        }
        lastNode = lastNode -> next;
    }
```

```
    temp = createNodeInDCLL();
    temp -> data = x;
    if (pos == 1) {
        if (first == NULL) {
            temp -> next = temp;
            temp -> prev = temp;
        } else {
            lastNode = first-> prev;
            temp -> prev = lastNode;
            temp -> next = first;
            lastNode -> next = temp;
            first -> prev = temp;
        }
        first = temp;
    } else {
        temp -> prev = lastNode;
        temp -> next = lastNode -> next;
        lastNode -> next = temp;
        temp -> next -> prev = temp;
    }

    return first;
}

int countInDCLL (NODE first) {
    NODE temp = first;
    int sum = 0;
    do {
        sum++;
        temp = temp -> next;
    } while(temp != first);
    return sum;
}

NODE deleteAtPositionInDCLL(NODE first, int pos) {
    NODE temp = first, lastNode = first;
    int i;
```

```
    if (pos == 1) {
        if (temp -> next == first) {
            first = NULL;
        } else {
            lastNode = first -> prev;
            lastNode -> next = first -> next;
            first -> next -> prev = lastNode;
            first = first -> next;
        }
    } else {
        for (i = 1; i < pos; i++) {
            if (temp -> next == first) {
                printf("No such position in Doubly Circular Linked List so deletion is not possible\n");
                return first;
            }
            lastNode = temp;
            temp = temp -> next;
        }
        if (temp -> next == first) {
            lastNode -> next = first;
            first -> prev = lastNode;
        } else {
            lastNode -> next = temp -> next;
            temp -> next -> prev = lastNode;
        }
    }

    printf("The deleted element from DCLL : %d\n",  temp -> data);
    free(temp);
    return first;
}
NODE deleteAtBeginInDCLL(NODE first) {
    NODE temp = first, lastNode = first;
    if(temp -> next == first) {
        first = NULL;
    } else {
```

```
        lastNode = first -> prev;
        first = first -> next;
        first -> prev = lastNode;
        lastNode -> next = first;
    }
    printf("The deleted element from DCLL : %d\n", temp -> data);
    free(temp);
    return first;
}
NODE deleteAtEndInDCLL(NODE first) {
    NODE lastNode, temp = first;
    if (temp -> next == first) {
        first = NULL;
    } else {
        temp = first -> prev;
        lastNode = temp -> prev;
        lastNode -> next = first;
        first -> prev = lastNode;
    }

    printf("The deleted element from DCLL : %d\n", temp -> data);
    free(temp);
    return first;
}

NODE deleteList(NODE head_ref)
{
    NODE current = head_ref;
    NODE next;
    do{


        next = current->next;
        free(current);
        current = next;
    }while(current!=head_ref);
    head_ref = NULL;
```

```
    return head_ref; }



int main() {
    NODE first = NULL;
    int select = 0,x,pos;
    printf("\t\tDOUBLE CIRCULAR LINKED LIST IMPLEMENTATION\n\n");
    do{
        printf("\t1.ADD NODES\n\t2.INSERT AT BEGIN\n\t3.INSERT AT END\n\t4.INSERT AT POSITION\n\t5.DELETE AT
BEGIN\n\t6.DELETE AT END\n\t7.DELETE AT POSITION\n\t8.COUNT\n\t9.TRAVERSE LIST\n\t10.SEARCH\n\t11.DELETE
LIST\n\t12.EXIT\n");


        printf("\tPlease Enter Your Choice\n");
        scanf("%d",&select);
        switch(select)
        {
            case 1:
                first = NULL;

                printf("Enter an element (Stops when you enter -1): ");
                scanf("%d", &x);
                while (x != -1) {
                    first = addNodesInDCLL(first, x);
                    printf("Enter an element : ");
                    scanf("%d", &x);
                }
                printf("-------------------------------------------------------\n");
                break;
            case 2:


                printf("Enter an element : ");
                scanf("%d", &x);
                first = insertAtBeginInDCLL(first, x);
```

```
    printf("------------------------------------------------------------\n");
    break;
case 3:


    printf("Enter an element : ");
    scanf("%d", &x);
    first = insertAtEndInDCLL(first, x);


    printf("------------------------------------------------------------\n");
    break;
case 4:


    printf("Enter a position : ");
    scanf("%d",&pos);
    printf("Enter an element : ");
    scanf("%d", &x);
    if (pos <= 0 || (pos > 1 && first == NULL)) {
        printf("No such position in Doubly Circular Linked"
               " List so insertion is not possible\n");
    } else {
        first = insertAtPositionInDCLL(first, pos, x);
    }

    printf("------------------------------------------------------------\n");
    break;
case 5:

    if(first==NULL){
        printf("Double Circular Linked List is empty so deletion is not possible\n");}
        else{
            first = deleteAtBeginInDCLL(first);
        }
```

```
    printf("------------------------------------------------------------\n");
    break;
case 6:
    if (first == NULL) {
        printf("Doubly Circular Linked List is empty "
               "so deletion is not possible\n");
    } else {
        first = deleteAtEndInDCLL(first);
    }



    printf("------------------------------------------------------------\n");
    break;
case 7:


    if (first == NULL) {
        printf("Doubly Circular Linked List is empty "
               "so Deletion is not possible");
    } else {
        printf("Enter a position : ");
        scanf("%d", &pos);
        if (pos <= 0) {
            printf("No such position in Doubly Circular Linked"
                   " List so deletion is not possible\n");
        } else {
            first = deleteAtPositionInDCLL(first, pos);
        }
    }

    printf("------------------------------------------------------------\n");
    break;
```

```
case 8:
    if (first == NULL) {
        printf("DCLL is empty.\n");
    } else {
        printf("The number of elements in the DCLL are : %d\n",countInDCLL(first));
    }


    printf("-----------------------------------------------------------\n");
    break;
case 9:
    if (first == NULL) {
        printf("Linked List is empty\n");
    } else {
        printf("The elements in Doubly Circular Linked"
                " List are : ");
        traverseListInDCLL(first);
    }


    printf("-----------------------------------------------------------\n");
    break;
case 10:


    printf("Enter the element to be searched: ");
    scanf("%d" ,&x);
    pos = searchPosOfEleInDCLL(first, x);
    if (pos == 0) {
        printf("The given element %d is not found in the DCLL\n",x);
                } else {
                    printf("The given element %d is found at the location: %d\n",x,pos);
                        }
```

```
printf("-------------------------------------------------------------\n");
break;
case 11:
first = deleteList(first);

printf("-------------------------------------------------------------\n");
break;

case 12:
break;

default:
printf("\t\n\nYou have not entered the right choice\n\n");

}
}while(select!=12);

}
```