POORNIMA

Experiment 8

Aim –: Write a program to perform following operation on circular linked list:

1) Insert at first location
2) Insert at last location
3) Delete from start
4) Delete from end
5) Traversing
6) Exit

Program –:

```
#include<iostream>
#include<stdlib.h>
using namespace std;

typedef struct node {
        int info;
        struct node *link;
} NODE;

struct node *start;
struct node *last;

int score=2;
```

```
void InsertFLCL(){
    NODE *ptr;
    ptr = (NODE *)malloc(sizeof(NODE));
    int item;
    cout<<"Enter item : ";
    cin>>item;
    ptr->info = item;
    if(start==NULL){
        start = ptr;
        ptr->link = start;
        last = start;
    }
    else{
        ptr->link = start;
        start = ptr;
        lsst->link = start;
    }
}
void InsertLLCL(){
    NODE *ptr;
    ptr = (NODE *)malloc(sizeof(NODE));
    int item;
    cout<<"Enter item : ";
    cin>>item;
    ptr->info = item;
    if(start==NULL){
        start = ptr;
        ptr->link = start;
```

Enter choice
1. Insert at first location
2. Insert at last location
3. Delete from start
4. Delete from end
5 Traversing
6. Exit
1
Enter item : 10
Enter choice
1. Insert at first location
2 Insert at last location
3. Delete from start
4 Delete from end
5. Traversing
6. Exit
2
Enter item : 20
Enter choice
1. Insert at first location
2. Insert at last location
3. Delete from start
4. Delete from end
5 Traversing
6. Exit
1
Enter item : 30

```
        last = start;
    }
    else {
        last -> link = ptr;
        last = ptr;
        last -> link = start;
    }
}
void Traversing () {
    if (start == NULL || score == 0) {
        cout << "Empty link list \n";
    }
    else {
        NODE *ptr;
        ptr = start;
        cout << "Circular Linked list \n Start ";
        do {
            cout << " -> " << ptr -> info;
            ptr = ptr -> link;
        } while (ptr != start);
        cout << " -> Start \n";
    }
}
void DeleteFLCL () {
    if (start == NULL || score == 0) {
        cout << "Empty link list";
    }
    else {
```

Enter choice
1. Insert at first location
2. Insert at last location
3. Delete from start
4. Delete from end
5. Traversing
6. Exit
2
Enter item : 40
Enter choice
1. Insert at first location
2. Insert at last location
3. Delete from start
4. Delete from end
5. Traversing
6. Exit
3
Deleted item is : 30
Enter choice
1. Insert at first location
2. Insert at last location
3. Delete from start
4. Delete from end
5. Traversing
6. Exit
4
Deleted item is : 40

```
int item;
item = start->info;
start = start->link;
last->link = start;
cout<<"Deleted item is : "<<item<<endl;}
if(start==last){
    score = score-1;
}
}
void DeleteLLCL(){
    if(start==NULL||score==0){
        cout<<"Empty link list\n";
    }
    else{
    int item;
    item = last->info;
    NODE *ptr;
    ptr = start;
    while(ptr->link!=last){
        ptr = ptr->link;
    };
    last =ptr;
    last->link = start;
    cout<<"Deleted item is : "<<item<<endl;}
    if(start==last){
        score = score-1;
    }
}
```

Enter choice
1. Insert at first location
2 Insert at last location
3. Delete from start
4. Delete from end
5. Traversing
6. Exit
5
Circular Linked list
Start → 10 → 20 → Start

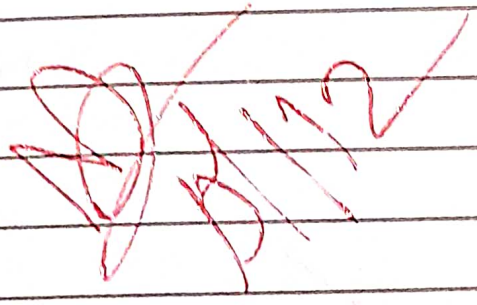
TYOHNIMA

```
int main () {
    start = NULL;
    last = NULL;
    int choice;
    do {
    cout<< "Enter choice "<<endl;
    cout<< "1 Insert at first location "<<endl;
    cout<< "2 Insert at last location "<<endl;
    cout<< "3 Delete from start "<<endl;
    cout<< "4 Delete from end "<<endl;
    cout<< "5 Traversing "<<endl;
    cout<< "6 Exit "<<endl;
    cin>> choice;

    switch(choice) {
      case 1: {
            InsertFLCL();
            break;
      }
      case 2: {
            InsertLLCL();
            break;
      }
      case 3: {
            DeleteFLCL();
            break;
      }
      Case 4: {
            DeleteLLCL();
```

```
        break;
    }
    case 5:{
        Traversing();
        break;
    }
    case 6:{
        break;
    }
    default :{
        break;
    }
}while(choice!=6);

return 0;
}
```



Queue Operations
Enter choice :
1 Insert in Queue
2 Delete from Queue
3. Traversing
4 Exit
1
Enter item : 10

Queue Operations
Enter choice :
1 Insert in Queue
2 Delete from Queue
3 Traversing
4 Exit
1
Enter item : 20

Queue Operations
Enter choice
1. Insert in Queue
2. Delete from Queue
3 Traversing
4 Exit
1
Enter item : 30

POORNIMA

Experiment 9

Aim -: Write a program to perform following operations on queue.

1) Insert in Queue
2) Delete from Queue
3) Traversing
4) Exit

i) Using Linked List

```
#include<iostream>
#include<stdlib.h>
using namespace std;

typedef struct node {
        int info;
        struct node* link;
}NODE;
NODE *front, *rear;

void QueueInsert() {
    NODE* ptr;
    ptr = (NODE*)malloc(sizeof(NODE));
    int item;
    cout<<"Enter item : ";
    cin>>item;
    ptr->info =item;
    if (rear==NULL) {
        rear = ptr;
```

Queue Operations
Enter choice:
1 Insert in Queue
2 Delete from Queue
3 Traversing
4 Exit
1
Enter item : 40

Queue Operations
Enter choice:
1 Insert in Queue
2 Delete from Queue
3 Traversing
4 Exit
3
Queue is : 10 20 30 40

Queue Operations
Enter choice:
1 Insert in Queue
2 Delete from Queue
3 Traversing
4 Exit
2
Deleted item is : 10

```
        front = rear;
        ptr -> link = NULL;
    }
    else {
        rear -> link = ptr;
        rear = ptr;
        ptr -> link = NULL;
    }
}
void QueueDelete() {
    int item;
    if (front == NULL) {
        cout << "Queue is Empty" << endl;
    }
    else {
        item = front -> info;
        front = front -> link;
        cout << "Deleted item is : " << item << endl;
    }
}
void QueueTraverse() {
    if (front == NULL) {
        cout << "Queue is Empty" << endl;
    }
    else {
        NODE *ptr;
        ptr = front;
        cout << "Queue is : ";
```

Queue Operations
Enter choice
1 Insert in Queue
2 Delete from Queue
3 Traversing
4 Exit
2
Deleted item is : 20

Queue Operations
Enter choice
1 Insert in Queue
2 Delete from Queue
3 Traversing
4 Exit
3
Queue is : 30 40

Queue Operations
Enter choice
1 Insert in Queue
2 Delete from Queue
3 Traversing
4 Exit
4

POORNIMA

```
while (ptr!=NULL){
    cout<<ptr->info<<" ";
    ptr=ptr->link;
  }
  cout<<endl;
 }
}
int main(){
 front=NULL;
 rear=NULL;
 int choice;
 do{
  cout<<"\n Queue Operations"<<endl;
  cout<<"Enter choice:"<<endl;
  cout<<"1 Insert in Queue"<<endl;
  cout<<"2 Delete from Queue"<<endl;
  cout<<"3 Traversing"<<endl;
  cout<<"4 Exit"<<endl;
  cin>>choice;
  switch(choice){
   case 1:{
     QueueInsert();
     break;}
   case 2:{
     QueueDelete();
     break;}
   case 3:{
     QueueTraverse();
     break;}
```

Page No.....

POORNIMA

```
Case 4: {
        break;
    }
default : {
        break;
    }
  }
} while (choice != 4);
return 0;
}
```

2) Using array

```
#include <iostream>
using namespace std;
int front = -1;
int rear = -1;
int queue[10];
int score = 1;
void QueueInsert () {
    score = 1;
    int item;
    cout << "Enter item : ";
    cin >> item;
    if (rear == -1) {
        queue[0] = item;
        front = front + 1;
        rear = rear + 1; }
```

```
else if(rear==10){
    cout<<"Queue full";
}
else{
    queue[rear+1]=item;
    rear=rear+1;
}
}
void QueueDelete(){
    if(front==-1||score==0){
        cout<<"Empty queue"<<endl;
    }
    else{
        if(front==rear){
            score=0;
        }
        int item;
        item=queue[front];
        front=front+1;
        cout<<"Deleted item is:"<<item;}
}
void QueueTraverse(){
    if(front==-1||score==0){
        cout<<"Empty queue"<<endl;
    }
    else{
        cout<<"Queue is:";
        for(int i=front;i<=rear;i++){
```

```
        cout<<queue[i]<<" ";
    }
    cout<<endl;
}
int main(){
    int choice;
    do{
    cout<<"\n Queue Operations "<<endl;
    cout<<"Enter choice :"<<endl;
    cout<<" 1 Insert in Queue "<<endl;
    cout<<" 2 Delete from Queue "<<endl;
    cout<<" 3 Traversing "<<endl;
    cout<<" 4 Exit "<<endl;
    cin>>choice;
    switch(choice){
      case 1:{
          QueueInsert();
          break;}
      case 2:{
          QueueDelete();
          break;}
      case 3:{
          QueueTraverse();
          break;
        }
      Case 4:{
          break;
        }
```

POORNIMA

```
        default : E
                break;
            }
        }
    } while (choice != 4);

    return 0;
}
```

POORNIMA

Experiment 10

Aim-: a) Write a program to search an element using linear search.
b) Wriote a program to search an element using binary search.

```
#include <iostream>
using namespace std;
int main() {
  cout<<"linear search\n";
  int n;
  cout<<"Enter size : ";
  cin>>n;
  int arr[n];
  cout<<"Enter elements :"<<endl;
  for(int i=0; i<n; i++){
      cin>>arr[i];
  }
  cout<<"Array elements are : ";
  for(int i=0; i<n; i++){
      cout<<arr[i]<<" ";
  }
  int item;
  cout<<"\n Enter item :";
  cin>>item;
  int loc=NULL;
  for(int i=0; i<n; i++){
```

POORNIMA

```
if(arr[i]==item){
    loc=i;
    cout<<"Item is present at index :"<<loc<<endl;
  }}
if(loc==NULL){
    cout<<"Item not present"<<endl;
  }
return 0;
}

#include<iostream>
using namespace std;
int main(){
  cout<<"Binary search\n";
  int n;
  cout<<"Enter size :";
  cin>>n;
  int arr[n];
  cout<<"Enter elements (sorted): "<<endl;
  for(int i=0; i<n; i++){
    cout<<arr cin>>arr[i];
  }
  cout<<"Array elements are :";
  for(int i=0; i<n; i++){
    cout<<arr[i]<<" ";
  }
  int item;
  cout<<"\n Enter item :";
```

POORNIMA

```
cin>>item;
int loc = NULL;
int beg=0, end=n-1, mid = int((beg+end)/2);
while(beg<end){
    if(arr[mid]==item){
        loc = mid;
        cout<<"Item is present at index : "<<loc;
        break;}
    else{
        if(arr[mid]<item){
            beg = mid+1;}
        else{
            end = mid-1;}
        mid = (beg+end+1)/2;
    }
}
if(loc==NULL){
    cout<<"Item not present"<<endl;
}
return 0;
}
```

Output →

```
Linear search
Enter size : 10
Enter elements :
2 40 25 12 34 10 11 98 87 100
Array elements are : 2 40 25 12 34 10 11 98 87 100
Enter item : 10
Item is present at index : 5
```

Output →

```
Binary search
Enter size : 10
Enter elements (sorted):
10 20 30 40 50 60 70 80 90 100
Enter item : 40
Item is present at index : 3
```