Module 4 : Queue

1) Queue using Linked List

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
#include<bits/stdc++.h>
using namespace std;
struct node{
        int data;
        struct node *next;
};

        struct node *front=NULL, *rear=NULL;

void insertion (int val)
{
                struct node *ptr;
                ptr=(struct node*)malloc(sizeof(struct node));
                ptr->data=val;
                ptr->next=NULL;
                if(front==NULL){
                        front = ptr;
                        rear=ptr;
                }
                else
                {
                        rear->next = ptr;
                        rear=ptr;
                }
}

void deletion()
{
        struct node *ptr;
        ptr=front;
        if(front==NULL)
        {
                cout<<"\n Queue s empty";
        }
        else
        {
                front=front->next;
                cout<<"\n Deleted element is"<<ptr->data;
                free(ptr);
```

```
        }
}

void display()
{
        struct node *ptr;
        ptr=front;
        if(front==NULL)
        {
                cout<<"\n Queue is empty";
        }
        else
        {
                cout<<"\n Elements in queue are :";
while(ptr!=NULL)
{
        cout<<ptr->data;
        ptr=ptr->next;
} cout<<endl;
//cout<<ptr->data;
        }
}
int main(){
                int val,option;
                cout<<"\nEnter your choice";

                do{

                        cout<<"\n1.Insert";
                        cout<<"\n2.Delete";
                        cout<<"\n3.Display";
                        cout<<"\n4.Exit";
                        cout<<"\nEnter your option : ";
                        cin>>option;
                        switch(option){
                                case 1:{
                                        cout<<"\nEnter the number to be inserted: ";
                                        cin>>val;
                                        insertion(val);
                                        break;
                                }
                                case 2:{
                                        deletion();
                                        break;
                                }
                                case 3:{
```

```
                display();
                break;
            }
            case 4:{
                exit(0);
                break;
            }
        }

    }while(option<=4);
        getch();
        return 0;

}
```

2) Circular Queue

```
#include <iostream>
#define MAX 5
using namespace std;
class Circular_Queue
{
   private:
      int *cqueue_arr;
      int front, rear;
   public:
      Circular_Queue()
      {
         cqueue_arr = new int [MAX];
         rear = front = -1;
      }
      void insert(int item)
      {
         if ((front == 0 && rear == MAX-1) || (front == rear+1))
         {
            cout<<"Queue Overflow \n";
```

```
        return;
    }
    if (front == -1)
    {
        front = 0;
        rear = 0;
    }
    else
    {
        if (rear == MAX - 1)
            rear = 0;
        else
            rear = rear + 1;
    }
    cqueue_arr[rear] = item ;
}
void del()
{
    if (front == -1)
    {
        cout<<"Queue Underflow\n";
        return ;
    }
    cout<<"Element deleted from queue is : "<<cqueue_arr[front]<<endl;
    if (front == rear)
    {
        front = -1;
        rear = -1;
    }
    else
```

```
    {
        if (front == MAX - 1)
            front = 0;
        else
            front = front + 1;
    }
}       void display()
{
    int front_pos = front, rear_pos = rear;
    if (front == -1)
    {
        cout<<"Queue is empty\n";
        return;
    }
    cout<<"Queue elements :\n";
    if (front_pos <= rear_pos)
    {
        while (front_pos <= rear_pos)
        {
            cout<<cqueue_arr[front_pos]<<"  ";
            front_pos++;
        }
    }
    else
    {
        while (front_pos <= MAX - 1)
        {
            cout<<cqueue_arr[front_pos]<<"  ";
            front_pos++;
        }
```

```
            front_pos = 0;
            while (front_pos <= rear_pos)
            {
               cout<<cqueue_arr[front_pos]<<"  ";
               front_pos++;
            }
         }
         cout<<endl;
      }
};
int main()
{
   int choice, item;
   Circular_Queue cq;
   do
   {
      cout<<"1.Insert\n";
      cout<<"2.Delete\n";
      cout<<"3.Display\n";
      cout<<"4.Quit\n";
      cout<<"Enter your choice : ";
      cin>>choice;
      switch(choice)
      {
      case 1:
         cout<<"Input the element for insertion in queue : ";
         cin>>item;
         cq.insert(item);
            break;
         case 2:
```

```
            cq.del();
               break;
         case 3:
            cq.display();
               break;
            case 4:
               break;
            default:
               cout<<"Wrong choice\n";
            }/*End of switch*/
      }
      while(choice != 4);
      return 0;
}
```

3) Double Ended Queue

```
#include<iostream>
using namespace std;
#define SIZE 5
class dequeue
{
   int a[10],front,rear;
   public:
      dequeue();
      void add_at_beg(int);
      void add_at_end(int);
      void delete_fr_front();
      void delete_fr_rear();
      void display();
};
```

```
dequeue::dequeue()
{
   front=-1;
   rear=-1;
}
void dequeue::add_at_end(int item)
{

   if(rear>=SIZE-1)
   {
      cout<<"\n insertion is not posibble,overflow!!!!";
   }
   else
   {
      if(front==-1)
      {
         front++;
         rear++;
      }
      else
      {
         rear=rear+1;
      }
   a[rear]=item;
   cout<<"\nInserted item is"<<a[rear];
   }
}
void dequeue::add_at_beg(int item)
{
```

```
        if(front==-1)
        {
           front=0;
           a[++rear]=item;
           cout<<"\n inserted element is"<<item;
        }
        else if(front!=0)
        {
           a[--front]=item;
           cout<<"\n inserted element is"<<item;

        }
        else
        {
             cout<<"\n insertion is not posibble,overflow!!!";
        }

}
void dequeue::display()
{
   if(front==-1)
   {
      cout<<"Dequeue is empty";
   }
   else
   {
      for(int i=front;i<=rear;i++)
      {
         cout<<a[i]<<" ";
      }
```

```
    }
}
void dequeue::delete_fr_front()
{
        if(front==-1)
        {
            cout<<"deletion is not possible::dequeue is empty";
            return;
        }
        else
        {
            cout<<"the deleted element is"<<a[front];
            if(front==rear)
            {
                front=rear=-1;
                return;
            }
            else
                front=front+1;
        }
}
void dequeue::delete_fr_rear()
{
        if(front==-1)
        {
            cout<<"deletion is not possible::dequeue is empty";
            return;
        }
        else
        {
```

```
        cout<<"the deleted element is"<<a[rear];
        if(front==rear)
        {
           front=rear=-1;
        }
        else
           rear=rear-1;
      }
}
int main()
{
   int c,item;
   dequeue d1;
   do
   {
      cout<<"\n\n***DEQUEUE OPERATION***\n";
      cout<<"\n 1_insert at beginning";
      cout<<"\n 2_insert at end";
      cout<<"\n 3_display";
      cout<<"\n 4_deletion from front";
      cout<<"\n 5_deletion from rear";
      cout<<"\n 6_exit";
      cout<<"\n enter your choice";
      cin>>c;
      switch(c)
      {
         case 1:cout<<"enter the element to be inserted";
                cin>>item;
                d1.add_at_beg(item);
                break;
```

```
        case 2:cout<<"enter the element to be inserted";
                cin>>item;
                d1.add_at_end(item);
                break;
        case 3:d1.display();
                break;
        case 4:d1.delete_fr_front();
                break;
        case 5:d1.delete_fr_rear();
               break;
        case 6:exit(1);
                break;
        csdefault:cout<<"invalid choice";
              break;
      }
   }
   while(c!=7);
}
```

4) Application of queue – Priority Queue

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cstdlib>
using namespace std;

/*
 * Node Declaration
 */
struct node
```

```
{
        int priority;
        int info;
        struct node *link;
};
/*
 * Class Priority Queue
 */
class Priority_Queue
{
   private:
      node *front;
   public:
      Priority_Queue()
      {
         front = NULL;
      }
      /*
       * Insert into Priority Queue
       */
      void insert(int item, int priority)
      {
         node *tmp, *q;
         tmp = new node;
         tmp->info = item;
         tmp->priority = priority;
         if (front == NULL || priority < front->priority)
         {
            tmp->link = front;
            front = tmp;
```

```
    }
    else
    {
        q = front;
        while (q->link != NULL && q->link->priority <= priority)
            q=q->link;
        tmp->link = q->link;
        q->link = tmp;
    }
}
/*
 * Delete from Priority Queue
 */
void del()
{
    node *tmp;
    if(front == NULL)
        cout<<"Queue Underflow\n";
    else
    {
        tmp = front;
        cout<<"Deleted item is: "<<tmp->info<<endl;
        front = front->link;
        free(tmp);
    }
}
/*
 * Print Priority Queue
 */
void display()
```

```
    {
        node *ptr;
        ptr = front;
        if (front == NULL)
            cout<<"Queue is empty\n";
        else
        {       cout<<"Queue is :\n";
            cout<<"Priority       Item\n";
            while(ptr != NULL)
            {
                cout<<ptr->priority<<"                "<<ptr->info<<endl;
                ptr = ptr->link;
            }
        }
    }
};
/*
 * Main
 */
int main()
{
    int choice, item, priority;
    Priority_Queue pq;
    do
    {
        cout<<"1.Insert\n";
        cout<<"2.Delete\n";
        cout<<"3.Display\n";
        cout<<"4.Quit\n";
        cout<<"Enter your choice : ";
```

```
        cin>>choice;
        switch(choice)
        {
        case 1:
            cout<<"Input the item value to be added in the queue : ";
            cin>>item;
            cout<<"Enter its priority : ";
            cin>>priority;
            pq.insert(item, priority);
            break;
        case 2:
            pq.del();
            break;
        case 3:
            pq.display();
            break;
        case 4:
            break;
        default :
            cout<<"Wrong choice\n";
        }
    }
    while(choice != 4);
    return 0;
}
```