```
/*WAP for list implementation of QUEUE*/
#include<iostream>
#include<stdlib.h>
#define max 10
using namespace std;
int avail = 0;
struct nodetype
{
   int info,next;
} node[max];
int getnode()
{
   int p;
   if(avail == -1)
   {
      cout<<"overflow";
      exit(1);
   }
   p = avail;
   avail = node[avail].next;
   node[p].next = -1;
   return(p);
}
void freenode(int p)
{
   node[p].next = avail;
   avail=p;
   return;
}
void display(int f)
{
   cout<<"\n\n============================================"<<endl;
   if(f!=-1)
   {
      cout<<"NODE"<<"\t\tINFO"<<endl;
      for(int i=f; i!=-1; i=node[i].next)
      {
            cout<<i<<"\t\t"<<node[i].info<<endl;
        }
    }
    else
    {
        cout<<"Empty Queue"<<endl;
    }
    cout<<"=============================================="<<endl;
}
class Queue
{
private:
    int front, rear;
public:
    Queue():front(-1),rear(-1) {};
    bool isempty()
    {
        if(front==-1)
            return true;
        else
            return false;
    }
    void enqueue()
    {
        int ptr;
        ptr = getnode();
        cout<<"Enter an integer:";
        cin>>node[ptr].info;
        if(rear==-1)
            front = ptr;
        else
            node[rear].next=ptr;
        rear=ptr;
    }
    int dequeue()
    {
        int delval,ptr;
        if(isempty())
        {
            cout<<"underflow";
            exit(1);
        }
        else
        {
            delval=node[front].info;
            ptr=front;
            front=node[front].next;
            if(front==-1)
                rear=-1;
            freenode(ptr);
            return delval;
        }
    }
    int getfront()
    {
        return front;
    }
};
int main()
{
    for(int i=0; i<max; i++)
    {
        if(i==max-1)
        {
            node[i].next=-1;
        }
        else
        {
            node[i].next=i+1;
        }
    }
    int choose=1;
    int val,pos;
    Queue Q;
```

```
   while(choose!=0)
   {
      cout<<"\nmenu:"<<endl;
      cout<<"=========="<<endl;
      cout<<"1 .Enqueue"<<endl;
      cout<<"2 .Dequeue"<<endl;
      cout<<"3 .Display Queue"<<endl;
      cout<<"0 .Exit"<<endl;
      cout<<"Enter your choice: ";
      cin>>choose;
      switch(choose)
      {
      case 1:
         Q.enqueue();
         display(Q.getfront());
         break;
      case 2:
         val=Q.dequeue();
         display(Q.getfront());
         cout<<val<<" is dequeued"<<endl;
         break;
      case 3:
         display(Q.getfront());
         break;
      case 0:
         break;
      }
   }
   return 0;
}

/*WAP for list implementation of QUEUE*/
#include<iostream>
#define max 4
using namespace std;
class Queue
{
   int avail;
   int front, rear;
   struct nodeType
   {
      int info, next;
   };
   struct nodeType node[max];
public:
   Queue()
   {
      avail=0;
      front=-1;
      rear=-1;
   }
   void initializelist()
   {
      for(int i=0; i<max; i++)
      {
         node[i].next=i+1;
         node[i].info=0;
      }
      node[max-1].next=-1;
   }
   int getnode()
   {
      int ptr;
      if (rear==max-1 && front==-1)
      {
         avail=0;
         rear=-1;
         initializelist();
      }
      ptr=avail;
      avail=node[ptr].next;
      return ptr;
   }
   void freenode(int p)
```

```
{
   node[p].next = avail;
   avail=p;
}
bool isfull()
{
   if (rear==max-1 && avail==-1)
   {
      cout<<"Overflow"<<endl;
      return true;
   }
   else
      return false;
}
bool isempty()
{
   if ((rear<front)||(front>max-1))
   {
      cout<<" Queue Underflow"<<endl;
      return true;
   }
   else
      return false;
}
void enqueue()
{
   int num,ptr;
   if (!(isfull()))
   {
      cout<<"Enter the number:\t";
      cin>>num;
      cout<<"\n";
      ptr=getnode();
      node[ptr].info=num;
      node[ptr].next=-1;
      //cout<<"Ptr value="<<ptr<<endl;
      cout<<node[ptr].info<<" is enqueued."<<endl;
      if (rear==-1)
      {
         front=ptr;
      }
      else
      {
         bool test=true;
         int temp=0;
         while(test)
         {
            if (node[temp].next==-1)
            {
               node[temp].next=ptr;
               test=false;
            }
            temp=node[temp].next;
         }
      }
      rear=ptr;
   }
}
void dequeue()
{
   int delval,ptr;
   if(!isempty())
   {
      delval= node[front].info;
      cout<<delval<<" is dequeued."<<endl;
      node[front].info=0;
      ptr = front;
      front = node[front].next;
      //if(front==-1)
      //  rear = -1;
      freenode(ptr);
   }
}
void display()
```

```
   {
      //cout<<"-------------------------------------"<<endl;
      cout<<"-------------------------------------"<<endl;
      cout<<"Front"<<front<<endl;
      cout<<"Rear="<<rear<<endl;
      cout<<"\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\tavail= "<<avail<<endl;
      cout<<"Node\t\t\t\t\tInfo\t\t\t\t\tNext"<<endl;
      for (int i=0; i<max; i++)
      {
         cout<<i<<"\t\t\t\t\t"<<node[i].info<<"\t\t\t\t\t"<<node[i].next<<endl;
      }
      cout<<"-------------------------------------"<<endl;
      //cout<<"-------------------------------------"<<endl;
   }
};
int main()
{
   int option;
   Queue qobj;
   qobj.initializelist();
   do
   {
      cout<<"Choose:\n1.Enqueue\n2.Dequeue\n3.Exit"<<endl;
      cin>>option;
      switch (option)
      {
      case 1:
         {
            qobj.enqueue();
            break;
         }
      case 2:
         {
            qobj.dequeue();
            break;
         }
      default:
         {
            cout<<"***************Exiting*************"<<endl;
         }
      }
      qobj.display();
   }
   while(option!=3);
   return 0;
}

/*WAP for list implementation of QUEUE*/
#include<iostream>
#define max 5
using namespace std;
//define a Queue //
template<class T>
class Queue
{
private:
   int front,rear;
   T arr[max];
   T sign;
public:
   // constructor to initialize front and rear
   Queue(T emptysign)
   {
      front=-1;
      rear=-1;
      sign=emptysign;
      for(int i= 0; i<max; i++)
      {
         arr[i]=sign;
      }
   }
   //isEmpty to check if queue is empty
   bool  isEmpty()
   {
    if (front == - 1)
    {
        return true;
    }
    else
    {
        return false;
    }
}
//to check if Queue is full
bool isFull()
{
    if (rear == max - 1)
    {
        return true;
    }
    else
    {
        return false;
    }
}
//enqueue into Queue
void enq(T data)
{
    if(!isFull())
    {
        if(front == -1)
            front = 0;
        arr[++rear] = data;
    }
    else
    {
        cout<<"Overflow"<<endl;
    }
}
//dequeue from the Queue
void deq()
```

```
    {
        if(!isEmpty())
        {
            cout << arr[front] << endl;
            arr[front++]=0;
            if(front>rear)
            {
                front=-1;
                rear=-1;
            }
        }
        else
        {
            cout<<"UnderFlow"<<endl;
        }
    }
    //display Queue
    void display()
    {
        cout<<"\n ==========="<<endl;
        cout<<"The queue is ==>\t";
        for(int i=0; i<max; i++)
        {
            cout<<arr[i]<<"\t";
        }
        if(front!=-1)
            cout<<"front:: "<<front%max<<"\tlen:: "<<rear-front+1<<endl;
        else
            cout<<"front:: "<<front%max<<"\tlen:: "<<rear-front<<endl;
        cout<<" ===========\n"<<endl;
    }
};
//driver main function
int main()
{
    Queue<int> que(0);
    char opt='a';
```

```
   int val;
   cout<<"what to do:\n"<<"d for dequeue:\n"<<"e for enqueue\n"<<"x for
display\n"<<"n for end"<<endl;
   while(opt!='n')
   {
      cout<<"your choice: ";
      cin>>opt;
      switch(opt)
      {
      case 'd':
         que.deq();
         break;
      case 'e':
         cout<<"enter value:";
         cin >> val;
         que.enq(val);
         break;
      case 'x':
         que.display();
         break;
      case 'n':
         cout<<"thank you"<<endl;
         break;
      }
   }
   return 0;
}

/*WAP for list implementation of QUEUE*/
#include<iostream>
#include<cstdlib>
#define MAX 15
using namespace std;
int avail =0;
struct nodetype
{
   int info,next;
```

```
};
class Queue
{
   nodetype node[MAX];
   int Front,rear;
   int getnode()
   {
      int p;
      if(avail==-1)
      {
         cout<<"\nOverflow\n\n";
         return -1;
      }
      p=avail;
      avail=node[avail].next;
      return p;
   }
   void freenode(int p)
   {
      node[p].info=-11; /** -11 denotes empty*/
      node[p].next=avail;
      avail=p;
   }
   bool Isempty()
   {
      if(rear==-1)
         return true;
      else
         return false;
   }
public:

   Queue():Front(-1),rear(-1)
   {
      for(int i=0; i<=MAX-1; i++)
      {
         node[i].info=-11;  /** -11 denotes empty*/
```

```
        node[i].next=i+1;
    }
    node[MAX-1].next=-1;
}
void enqueue(int num)
{
    int ptr;
    ptr = getnode();
    if(ptr==-1)
        cout<<"\nThere is no available node\n";
    else
    {
        node[ptr].info=num;
        node[ptr].next = -1;
        if(rear== -1)
            Front=rear= ptr;
        else
            node[rear].next = ptr;
        rear = ptr;
    }
}
void dequeue()
{
    if(Isempty())
        cout<<"\nQUEUE Underflow\n";
    else
    {
        int delval,ptr;
        delval = node[Front].info;
        cout<<"\nThe dequeued element is : "<<delval<<endl;
        ptr = Front;
        Front = node[Front].next;
        if(Front==-1)
            rear = -1;
        freenode(ptr);
    }
}
```

```
    void displayqueue()
    {
        if(Isempty())
            cout<<"\nQUEUE Underflow\n";
        else
        {
            int point=Front;
            cout<<"\nThe queue is:\n";
            while(node[point].next!=-1)
            {
                cout<<node[point].info<<"\t";
                point=node[point].next;
            }
            cout<<node[point].info<<"\n\n";
        }
    }
    void displayarr()
    {
        cout<<"\n\nIndex\tValue\tNext\n";
        for(int i=0; i<MAX; i++)
        {
            cout<<i<<"\t"<<node[i].info<<"\t"<<node[i].next<<endl;
        }
        cout<<endl;
    }
};
int main()
{
    Queue q;
    int choice,num;
    while(1)
    {
        cout<<"1. Enqueue\n2. Dequeue\n3. Display queue\n4. Display array\n5.
Exit\nEnter your choice : ";
        cin>>choice;
        switch(choice)
        {
```

```
        case 1:
        {
           while(1)
           {
              int num;
              cout<<"\nEnter -1 to finish enqueue\nEnter the value:  ";
              cin>>num;
              if(num==-1)
                 break;
              q.enqueue(num);
           }
           break;
        }
        case 2:
        {
           q.dequeue();
           break;
        }
        case 3:
        {
           q.displayqueue();
           break;
        }
        case 4:
        {
           q.displayarr();
           break;
        }
        default :
           exit(0);
        }
     }
  }
```