Question1.cpp

```cpp
#include<iostream>
using namespace std;
class Queue { // A class to represent a queue
    private:
        int front=-1, rear=-1, size;
        unsigned capacity; //assuming capacity is the number of items currently in queue.
        int* array;
    public:
        Queue(int siz) { // constructor
            array = new int[siz];
            size=siz;
            for(int i=0;i<size;i++){
                array[i]=0;
            }
            capacity=0;
        }
        void insert(int j) {
            if(!isFull()){
                array[++rear%size]=j;
                if(isEmpty()){
                    front=0;
                }
                capacity++;
                cout<<"Value inserted successfully\n";
            }
            else{
                cout<<"Array is Full, returning..."<<endl;
            }
        }
        int remove() {
            if(!isEmpty()){
                int  temp=array[front];
                front++;
                front = front%size;
                    capacity--;
                return temp;
                cout<<"Value removed successfully\n";
            }
            else{
                cout<<"Empty Array, returning..."<<endl;
            }
        }
        int peek() { //equivalent to accessing first element
            return array[front];
        }
        bool isEmpty() {
            return capacity==0;
        }
        bool isFull() {
            return capacity==size;
        }
        int Size() {
            return size;
```

```
        }

        void display(){
            if(!isEmpty()){
                cout<<"[FRONT]\t";
                for(int i=0;i<capacity;i++){
                    cout<<array[(front+i)%size]<<"\t";
                }
                cout<<"[REAR]\nNumber of Items in Queue: "<<capacity<<endl;
            }
            else{
                cout<<"Queue is Empty\n";
            }
        }
};

int main(){
    cout<<"Enter Size of Array to Implement Queue: ";
    int s;
    cin>>s;
    while(s<=0){
        cout<<"Invalid size, please try again: ";
        cin>>s;
    }
    cout<<"\n Creating Queue of size "<<s<<" : \n";
    Queue q(s);
    do{
        cout<<"\nWhat would you like to do with the Queue?\n1.
Enqueue\t2.Dequeue\t3.Peek\t4.Display\n99.EXIT\n[Anything Else Defaults to Display]\n";
        cin>>s;
        if(s==1){
            cout<<"Enter Value to Insert: ";
            cin>>s;
            q.insert(s);
            s=1;
        }
        else if(s==2){
            q.remove();
        }
        else if(s==3){
            cout<<"Value Currently at front : "<<q.peek()<<endl;
        }
        else if(s==4){
            q.display();
        }
    }while(s!=99);
}
```

Question2.cpp

```cpp
#include <iostream>
#include "Queue.h"
using namespace std;

int main()
{
    Queue<int> *q = new Queue<int>(10);
    if (q->isEmpty())
    {
        cout << "Queue is empty." << endl;
    }
    q->Put(1);
    q->Put(2);
    q->Put(3);
    while (!q->isEmpty())
    {
        cout << q->Get() << endl;
    }
    return 0;
}
```

Queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

#include <iostream>
using namespace std;

template <class T>
class Node
{
public:
    T data;
    Node *next;
    Node(T data)
    {
        this->data = data;
        next = nullptr;
    }
};

template <class T>
class Queue
{
private:
    Node<T> *rear;
    Node<T> *front;
    int size;
    int capacity;

public:
    Queue(int capacity) : rear(nullptr), front(nullptr)
    {
        this->capacity = capacity;
        size = 0;
    }

    bool isEmpty()
    {
        return (rear == nullptr && front == nullptr);
    }

    bool isFull()
    {
        return (size == capacity);
    }

    void Put(T value)
    {
        Node<T> *newnode = new Node<T>(value);
        if (isFull())
        {
            cerr << "Cannot enqueue queue is full" << endl;
            delete newnode;
            return;
```

```
54          }
55          else if (isEmpty())
56          {
57              rear = front = newnode;
58          }
59          else
60          {
61              rear->next = newnode;
62              rear = newnode;
63              ++size;
64          }
65      }
66
67      void dequeue()
68      {
69          if (isEmpty())
70          {
71              cerr << "Queue is empty cannot dequeu." << endl;
72              return;
73          }
74          else if (rear == front)
75          {
76              rear = front = nullptr;
77          }
78          else
79          {
80              Node<T> *temp = front;
81              front = front->next;
82              delete temp;
83          }
84          --size;
85      }
86
87      T Get()
88      {
89          T value = front->data;
90          dequeue();
91          return value;
92      }
93  };
94  #endif
```

Question3.cpp

```cpp
#include <iostream>
using namespace std;

template <class T>
class Queue
{
private:
    int front, rear;
    unsigned capacity;
    T *array;

public:
    Queue(int capacity)
    {
        array = new T[capacity];
        front = -1;
        rear = -1;
        this->capacity = capacity;
    }

    bool isEmpty()
    {
        return (rear == -1 && front == -1);
    }

    bool isFull()
    {
        return ((rear + 1) % capacity == front);
    }

    void enqueue(T element)
    {
        if (isFull())
        {
            cerr << "Queue is full cannot enqueue." << endl;
            return;
        }
        else if (isEmpty())
        {
            front = rear = 0;
        }
        else
        {
            rear = (rear + 1) % capacity;
        }
        array[rear] = element;
        cout << array[rear] << " is enqueued in the queue." << endl;
    }

    void dequeue()
    {
        if (isEmpty())
        {
```

```
54              cerr << "Queue is Empty cannot dequeue." << endl;
55              return;
56          }
57          else if (front == rear)
58          {
59              cout << array[front] << " is dequeued from the queue." << endl;
60              rear = front = -1;
61          }
62          else
63          {
64              cout << array[front] << " is dequeued from the queue." << endl;
65              front = (front + 1) % capacity;
66          }
67      }
68
69      T peek()
70      {
71          if (isEmpty())
72          {
73              cerr << "Queue is empty." << endl;
74          }
75          return array[front];
76      }
77
78      int queue_size()
79      {
80          if (isEmpty())
81              return 0;
82          else if (rear >= front)
83          {
84              return rear - front + 1;
85          }
86          else
87          {
88              return (capacity - front + rear) % capacity;
89          }
90      }
91  };
92
93  int main()
94  {
95      Queue<char> q(10);
96      q.enqueue('a');
97      q.enqueue('b');
98      q.enqueue('c');
99      q.enqueue('d');
100     q.enqueue('e');
101     q.enqueue('f');
102     q.enqueue('g');
103     q.enqueue('h');
104     q.enqueue('h');
105     q.enqueue('h');
106     q.enqueue('h');
107     q.enqueue('h');
108     q.enqueue('h');
109     q.dequeue();
```

```
110     q.dequeue();
111     cout << "Front element " << q.peek() << endl;
112     cout << "Size of queue " << q.queue_size() << endl;
113     return 0;
114 }
115
```