```
/*******************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1C
 * SECTION    : MW 5pm
 * Assign #8  : Templates
 * DUE DATE   : 25 March 2020
 ******************************************************************************/

#ifndef MYHEADER_H_
#define MYHEADER_H_

//Preprocessor directives

#include<iostream>  //for input and output

//using the name space standard
using namespace std;

//template class for Queue
template <class X>

//class Queue: class with attributes of the queues
class Queue
{
private:
    X *a;
    int frnt,rear;
    int size;
    int maxSize;

public:
//constrcutor
    Queue(int n)
    {
        a=new X[n];
        maxSize=n;
        frnt =0;
        rear=-1;
        size=0;
    }

    //function to insert element in queue
    void enqueue(X value)
    {
        if(isFull())
        {
            cout<<"Queue is full... Can't insert'\n";
            return;
        }
        rear=(rear+1)%maxSize;
        a[rear]=value;
        cout<<"Inserted element "<<value<<"\n";
        size++;
    }


    //function to remove element from queue
    X dequeue()
    {
        X temp;
        if(isEmpty())
        {
            cout<<"Queue is empty.....\n";
            return temp;
        }

        temp=a[frnt];
        frnt=(frnt+1)%maxSize;
        size--;
        cout<<"Removed element "<<temp<<"\n";
        return temp;
    }


    //function to get the front element from queue
    X front()
    {
        if(isEmpty())
        {
            cout<<"Queue is empty...\n";
        }
        return a[frnt];
    }


    //function to check if the queue is full
    bool isFull()
    {
        if(size==maxSize)
            return true;
        else
            return false;
    }


    //function to check is queue is empty
    bool isEmpty()
    {
        if(size==0)
            return true;
        else
            return false;
    }


    //function to get the size of the queue
    int Size()
    {
        return size;
    }
};



#endif /* MYHEADER_H_ */

```

```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1C
 * SECTION    : MW 5pm
 * Assign #8  : Templates
 * DUE DATE   : 25 March 2020
 ****************************************************************************/

#include"MyHeader.h"

int main(){
    /*****************************************************************
     * Queuing and deleting the objects to the stacks
     * _______________________________________________________________
     * using the templates and the functions, the user can queue or
     * delete objects from the head of the stacks and delete from the
     * end of the stack. Also, using the class function, the program
     * determines if the stack is empty or full
     * _______________________________________________________________
     * INPUT: N/A
     *
     * OUTPUT: outputs step by step of adding the objects the objects
     *         or deleting the objects from the stack, also outputs
     *         if the
     *
     *****************************************************************/


    cout << "/*****************************************************************\n"
         << " * Queuing and deleting the objects to the stacks\n"
         << " * _______________________________________________________________\n"
         << " * using the templates and the functions, the user can queue or\n"
         << " * delete objects from the head of the stacks and delete from the\n"
         << " * end of the stack. Also, using the class function, the program\n"
         << " * determines if the stack is empty or full\n"
         << " * _______________________________________________________________\n"
         << " * INPUT: N/A\n"
         << " *\n"
         << " * OUTPUT: outputs step by step of adding the objects the objects\n"
         << " *         or deleting the objects from the stack, also outputs\n"
         << " *         if the\n"
         << " *\n"
         << "*****************************************************************/\n\n";

//creating a Character queue
Queue<string> q(10);

//Queuing the string characters to the head of the stack
q.enqueue("a");
q.enqueue("b");
q.enqueue("c");
q.enqueue("d");
q.enqueue("e");
q.enqueue("f");
//deleting the string characters from buttom of the stack
q.dequeue();
q.dequeue();
q.dequeue();
```

```
//Queuing the string characters to the head of the stack
q.enqueue("g");
q.enqueue("h");
q.enqueue("i");
q.enqueue("j");
//deleting the string characters from buttom of the stack
q.dequeue();
q.dequeue();
q.dequeue();
q.dequeue();
q.dequeue();
q.dequeue();
q.dequeue();
q.dequeue();
cout<<"Front of queue: "<<q.front()<<"\n";


//creating an integer queue
Queue<int> d(10);

//Queuing the integers to the head of the stack
d.enqueue(1);
d.enqueue(2);
d.enqueue(3);
d.enqueue(4);
d.enqueue(5);
d.enqueue(6);
//deleting the integers from buttom of the stack
d.dequeue();
d.dequeue();
//Queuing the integers to the head of the stack
d.enqueue(7);
d.enqueue(8);
d.enqueue(9);
//deleting the integers from buttom of the stack
d.dequeue();
d.dequeue();
cout<<"Front of queue: "<<d.front()<<"\n\n";


//creating an Double queue
Queue<double> i(10);

//Queuing the doubles to the head of the stack
i.enqueue(1.1);
i.enqueue(2.1);
i.enqueue(3.3);
i.enqueue(4.4);
i.enqueue(5.5);
i.enqueue(6.6);
//deleting the doubles from buttom of the stack
i.dequeue();
//Queuing the doubles to the head of the stack
i.enqueue(7.7);
i.enqueue(8.8);
//deleting the doubles from buttom of the stack
i.dequeue();
i.dequeue();
i.dequeue();
```

```cpp
i.dequeue();
i.dequeue();
cout<<"Front of queue: "<<i.front()<<"\n\n";


return 0;
}
```