```
/*********************************************************************************
 * 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()
```

```
60      {
61          X temp;
62          if(isEmpty())
63          {
64              cout<<"Queue is empty.....\n";
65              return temp;
66          }
67
68          temp=a[frnt];
69          frnt=(frnt+1)%maxSize;
70          size--;
71          cout<<"Removed element "<<temp<<"\n";
72          return temp;
73      }
74
75
76      //function to get the front element from queue
77      X front()
78      {
79          if(isEmpty())
80          {
81              cout<<"Queue is empty...\n";
82          }
83          return a[frnt];
84      }
85
86
87      //function to check if the queue is full
88      bool isFull()
89      {
90          if(size==maxSize)
91              return true;
92          else
93              return false;
94      }
95
96
97      //function to check is queue is empty
98      bool isEmpty()
99      {
100         if(size==0)
101             return true;
102         else
103             return false;
104     }
105
106
107     //function to get the size of the queue
108     int Size()
109     {
110         return size;
111     }
112 };
113
114
115
116 #endif /* MYHEADER_H_ */
117
```

```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();
```

```
60 //Queuing the string characters to the head of the stack
61 q.enqueue("g");
62 q.enqueue("h");
63 q.enqueue("i");
64 q.enqueue("j");
65 //deleting the string characters from buttom of the stack
66 q.dequeue();
67 q.dequeue();
68 q.dequeue();
69 q.dequeue();
70 q.dequeue();
71 q.dequeue();
72 q.dequeue();
73 q.dequeue();
74 cout<<"Front of queue: "<<q.front()<<"\n";
75
76
77 //creating an integer queue
78 Queue<int> d(10);
79
80 //Queuing the integers to the head of the stack
81 d.enqueue(1);
82 d.enqueue(2);
83 d.enqueue(3);
84 d.enqueue(4);
85 d.enqueue(5);
86 d.enqueue(6);
87 //deleting the integers from buttom of the stack
88 d.dequeue();
89 d.dequeue();
90 //Queuing the integers to the head of the stack
91 d.enqueue(7);
92 d.enqueue(8);
93 d.enqueue(9);
94 //deleting the integers from buttom of the stack
95 d.dequeue();
96 d.dequeue();
97 cout<<"Front of queue: "<<d.front()<<"\n\n";
98
99
100 //creating an Double queue
101 Queue<double> i(10);
102
103 //Queuing the doubles to the head of the stack
104 i.enqueue(1.1);
105 i.enqueue(2.1);
106 i.enqueue(3.3);
107 i.enqueue(4.4);
108 i.enqueue(5.5);
109 i.enqueue(6.6);
110 //deleting the doubles from buttom of the stack
111 i.dequeue();
112 //Queuing the doubles to the head of the stack
113 i.enqueue(7.7);
114 i.enqueue(8.8);
115 //deleting the doubles from buttom of the stack
116 i.dequeue();
117 i.dequeue();
118 i.dequeue();
```

```
119 i.dequeue();
120 i.dequeue();
121 cout<<"Front of queue: "<<i.front()<<"\n\n";
122
123
124 return 0;
125 }
126
127
128
129
130
```