Student name :	Carter Hawks
Student email :	ckh170000@utdallas.edu
Class name :	2336.001_F18
Submitted on :	Nov 04, 2018 10:10 pm

MyQueueDriver.cpp

```
/*
 * MyQueueDriver.cpp
 *
 */

#include <iostream>
#include <string>
#include "MyQueue.cpp"
using namespace std;

int main() {
        MyQueue q;
        int which = 0;
        int quantity = 0;
        cin >> which;
        if (which != 1)
                cin >> quantity;
        string elements[quantity];
        for (int i = 0; i < quantity; i++)
                cin >> elements[i];
        switch (which) {
                case 1 : // test empty queue
                        cout << q.toString() << endl;
                        break;
                case 2 : // test insert method
                        for (string s : elements)
                                q.insert(s);
                        cout << q.toString() << endl;
                        break;
                case 3 : // test remove method
                        for (string s : elements)
                                q.insert(s);
                        q.remove();
                        cout << q.toString() << endl;
                        break;
        }
}
```

MyQueue.cpp

```
#include<cstdio>
#include <iostream>
#include <stack>
#include <string>
using namespace std;

/*
Analysis
We have to make a queue (LIFO) out of two stacks (each FIFO).
```

```
Design
size() - this is the two stacks sizes' added together
insert() - if the value we want to insert is null, an empty string,
or space, do not add it. Otherwise, push it to the first stack.
remove() - shiftstacks() until stack1 is empty, and then pop the first element on the 2nd stack and return it.
isEmpty() - if stack1 size + stack2 size is 0, empty is true
isFull() - if stack1 size + stack2 size >= maxCapacity, then full is true
shiftstacks() - pop element from stack1 and push to stack2 if stack1 has content

*/

class MyQueue {
        private:
                int maxCapacity = 4;
                stack<string> stack1;
                stack<string> stack2;

                void shiftstacks() {
            // Write your code here
            if(size() > 0 && stack1.size() > 0){
                string s = stack1.top();
                stack1.pop();
                stack2.push(s);
            } else {
                // do nothing pls
            }

                }

        public:
                MyQueue() {}
                ~MyQueue() {}

                int size() {
                        // Write your code here
            return stack1.size() + stack2.size();
                }

        // assuming this means "add to queue"
                void insert(string value) {
                         // Write your code here

            if(isFull()){
                // do nothing
            } else {
                // Your Queue should not accept null or empty String or space as an input
                if(value == "" || value.empty() || value == " "){
                    // do nothing
                } else {
                    stack1.push(value);
                }
            }

                }

        // assuming this means pop
                string remove() {
            if(isEmpty()){
                return "";
            } else {
                // Write your code here
                if(stack1.size() != 0){
                    while(stack1.size() != 0){
                        shiftstacks();
                    }
                }

                string top = stack2.top();
                stack2.pop();
                return top;
            }

                }

                bool isEmpty() {
            return (stack1.size() + stack2.size() == 0);
```

```
    }

    bool isFull() {
return (stack1.size() + stack2.size() >= maxCapacity);
    }

    //[QueueSize:Full/Empty:QueueElementsList]
    string toString() {
            //shiftstacks();
            string sb;
            sb.append("[");
            sb.append(to_string(this->size())).append(":");
            if (this->isEmpty())
                    sb.append("Empty").append(":");
            else if (this->isFull())
                    sb.append("Full").append(":");
            while (!this->isEmpty()) {
                    sb.append(this->remove());
                    if(this->size() != 0) sb.append(", ");
            }
            sb.append("]");
            return sb;
    }
};
```

Name

Custom test case

Input

2 3 a b c

Output (Lines:1)

[3:a, b, c]

Expected Output (Lines:0)

Status

NA

Name

Custom test case

Input

3 4 a b c d

Output (Lines:1)

[3:b, c, d]

Expected Output (Lines:0)

Status

NA

Name

Custom test case

Input

2 4 a b c d

Output (Lines:1)

[4:Full:a, b, c, d]

Expected Output (Lines:0)

Status
NA

Name
empty queue

Input
1

Output (Lines:1)
[0:Empty:]

Expected Output (Lines:1)
[0:Empty:]

Status
Pass

Name
insert method1

Input
2 2 a b

Output (Lines:1)
[2:a, b]

Expected Output (Lines:1)
[2:a, b]

Status
Pass

Name
remove method1

Input
3 2 a b

Output (Lines:1)
[1:b]

Expected Output (Lines:1)
[1:b]

Status
Pass
