Exercise – Queues and Dequeues

Exercise 1:

Create a queue class using the layout provided in the slides. It should contain the following functionality:

1.	Push	– Pushes a element onto the end of the queue
2.	Pop	– Removes the element at the front of the queue
3.	Top	- Get the element at the front of the queue
4.	IsEmpty	- Returns true if the queue is empty
5.	GetSize	- Returns the number of elements in the queue

Exercise 2:

You are now going to use your queue class to simulate a server handling messages sent to it by a client. Due to CPU constraints, your server can only handle a maximum of 5 messages per frame.

First, download TreasureHunt.zip from Portal, and extract it.

Using the example code in TreasureHunt.cpp and using your queue class, add functionality to the TreasureHunt.cpp main function to allow it to queue messages if there are more than 5 messages received from the client in a single frame. Your applications output should look similar to the output below:

```
Size of Message Queue before Retrieving Messages:  8

...

Size of Message Queue after Retrieving Messages:  3

Size of Message Queue before Retrieving Messages: 3

...

Size of Message Queue after Retrieving Messages: 0

Player 3 found the treasure at position 61, 22!
```