NATIONAL UNIVERSITY
OF COMPUTER & EMERGING SCIENCES
PESHAWAR CAMPUS

Problem Set:	Assignment: A05	**Semester**:	Spring 2019
Points:	*10*		
Date Set:	*See SLATE*	**Due Date**:	*See SLATE*
Course:	CS217 - OOP	**Instructor**:	Dr. Nauman

1 Queue Writers and Queue Readers

We've already implemented a queue. Now we are going to put it to use. This is a very common case study in the real world that you have some entity that puts things to be processed in a queue and another entity takes stuff from the queue and processes it. The first entity is called a Writer (since it writes to the queue) and the second is called Reader (because it reads stuff from the queue).

1.1 Tasks 1: Make Queue Generic

The first thing you need to do is to take the implementation of the class Queue we did earlier and generalize it using class templates. After you're done, the Queue class should be able to handle either `int` or `string` type values. The rest of the FIFO logic should, of course, remain the same.

1.2 Task 2: The Writer

In this task, you need to create a separate class called `Writer`. This class should have one public function with the following signature:

```
void process_file(string filename, Queue<string> *q);
```

This function reads the contents of the file (given as `filename`) one line at a time. The line that is read should be *enqueued* in `q`. That's all this function does!

1.3 Task 3: The Reader

For this task, create yet another class called `Reader` that has one public function as follows:

```
void process_queue(Queue<string> *q);
```

The logic for this function is that it should continue to *dequeue* from `q` until there is nothing to dequeue any more. Whenever it dequeues a string, it should simply output its length. For example:

```
String:  [Abstraction] has length:  11
```

(Recall that whenever the queue is empty and we try to dequeue from it, we get an exception.)

See the starter file for understanding how the Queue, Reader and Writer classes are instantiated and how their functions are invoked.

2 Submission

After you're done with your class design, write down (**read: assignment must be hand-written**) the full code you've created and submit that on or before the deadline. Late assignments (or soft copies) will not be accepted.