CSCE 221 Cover Page
Programming Assignment #1
Due Date: Friday October 4, 2019

Submit this cover page along with your report

First Name: **Last Name**: **UIN:**

Any assignment turned in without a fully completed cover page will NOT BE GRADED.
Please list all below all sources (people, books, webpages, etc) consulted regarding this assignment:

CSCE 221 Students	Other People	Printed Material	Web Material (URL)	Other
1.	1.	1.	1.	1.
2.	2.	2.	2.	2.
3.	3.	3.	3.	3.
4.	4.	4.	4.	4.
5.	5.	5.	5.	5.

Recall that University Regulations, Section 42, define scholastic dishonesty to include acquiring answers from any unauthorized source, working with another person when not specifically permitted, observing the work of other students during any exam, providing answers when not specifically authorized to do so, informing any person of the contents of an exam prior to the exam, and failing to credit sources used. Disciplinary actions range from grade penalties to expulsion. Please consult the Aggie Honor System Office for additional information regarding academic misconduct – it is your responsibility to understand what constitutes academic misconduct and to ensure that you do not commit it.

I certify that I have listed above all the sources that I consulted regarding this assignment, and that I have not received nor given any assistance that is contrary to the letter or the spirit of the collaboration guidelines for this assignment.

Today's Date:

Printed Name (in lieu of a signature):

Stacks and Linked Lists

Description:
For this programming assignment, you will implement a **Stack** whose **size can grow** as elements are inserted into the stack. You will **build three different implementations** of this Stack. **Two** of these implementations will be **array-based**. Each of these two should take an **initial capacity** for the stack in the **constructor**. The only difference between these implementations will be what happens when the Stack is full. For the first implementation, **ArrayStack**, you should increase the size of the array by a **constant amount.** For the second implementation, **DoublingArrayStac**k, you should **double the size** of the array. Finally, for the **third implementation**, you should implement a Stack using a **Linked List.**

You will now **time your three implementations**. For each version of your stack, push a large amount of random numbers onto the stack (e.g. 1,000,000). Time how long this takes using the StopWatch class we provided with the assignment. However, you should measure the total time at **fixed intervals** (e.g. 10000 push operations). For the array-based implementations, you should start with some small capacity significantly smaller than the maximum number of push operations. For the ArrayStack, your increment amount should also be significantly smaller than the number of push operations. You will then **graph the times for the three implementations**. This process will allow you to see how your choice of implementation can affect performance.

Coding Portion (50 Points):

- Start with the following template: Stack.h, and create three versions of the Stack (using templates for the type of object) filling in all of the member functions.
- For the Linked List, you will need to implement another class for the nodes in the Linked List.
- Be sure to test the correctness of your algorithms and implementations. Ensure all stack operations of stack ADT (push, pop, top, size, isEmpty) are implemented and tested to be working as expected.
- Your code will be graded based on whether or not it compiles, runs, produces correct output, whether or not your experimental setup is correct, and your coding style (does the code follow proper indentation/style and comments).

Report (50 Points):
You will write a brief report that includes theoretical analysis, a description of your experiments, and discussion of your results. At a minimum, your report should include the following sections:

1. *Introduction*. In this section, you should describe the objective of this assignment.
2. *Theoretical Analysis*. In this section, you should provide an analysis of the complexity of a push operation. Describe the effect of a push operation and the advantages and disadvantages of the three strategies. What is the complexity of a push (on average) for the different implementations? What is the worst case complexity for the different implementations? Here, you should be able to leverage the material presented in the classroom, and also as referenced in the textbook.
3. *Experimental Setup*. In this section, you should provide a description of your experimental setup, which includes but is not limited to
   a. Machine specification
   b. How did you generate the test inputs? What input sizes did you test? Why? What parameters did you use for your initial Stack size and increment amount in the case of the ArrayStack?
   c. How many times did you repeat each experiment?
4. *Experimental Results*. In this section, you should compare the performance (running time) of the push() operation in the three different implementations to one another and to their theoretical complexity.

a. Make a plot showing the running time (y-axis) vs. the number of push operations (x-axis). You must use some electronic tool (matlab, gnuplot, excel, …) to create the plot. **Hand-written plots will NOT be accepted.**

b. Provide a discussion of your results, which includes but is not limited to:
   i. Which of the three Stack implementations performs the best? Does it depend on the input? Provide some reasoning behind your observation.
   ii. To what extent does the theoretical analysis agree with the experimental results? Attempt to understand and explain any discrepancies you note.