CSC 415 Spring 2014

Exercise – Dynamic Arrays

Due: Tuesday April 8, 2014 by 12 noon (before class)
Grade: 6 points; Bonus 2 points
(In the Assignments Category)

Student Names: ______________________________

Objective:
The purpose of this assignment is to develop a better understanding of and proficiency with implementing dynamic memory management in C++.

Instructions:
You will work on this exercise in pairs.

On Canvas, join the group assigned to you. Each group should make one complete submission.

If you do not join the right group, the grade cannot be assigned in Canvas.

Use terminal commands to interact with the OS.

Create a new subdirectory for this exercise and use a code-aware editor (e.g. XCode or Eclipse).

Download the code provided on Canvas under "Exercise – Dynamic Arrays" and use it for this exercise.

The files in this directory contain the definition of the `Stack` class implemented using a fixed size array, and a simple driver.

Modify the class so that it now uses a **dynamic** array. Start with an initial size of 5.

Document all your source code appropriately. Be sure to include all partners' names in all source files.

<u>Question 1</u>: **Since the `Stack` class dynamically manages memory usage, what other methods should be implemented to ensure that the class is "safe"? For each method, explain your rationale.**

Implement the methods, described in Question 1, along with any others that might be needed.

Save your code and use the terminal commands to compile your program using g++.

```
$ g++ Stack.cpp stackDriver.cpp –o stackExec
```

Examine the output to check that it is as expected. If it is not, debug your code, fix any errors or warnings, recompile, and test the program.

```
$ ./stackExec
```

Now create a script file to demonstrate that your program compiles and executes without errors:

```
$ script stackIN.script
```

where *IN* represents both / all partners' initials.

Compile the program again, and run it to demonstrate that it compiles without errors and works correctly.

Exit the script:

```
$ exit
```

Question 2: Draw a diagram to show the memory allocation at the point where stackOne has just been instantiated, i.e. right after the following statement in the driver.

```
Stack stackOne;
```

Question 3: Draw a diagram to show the memory allocation at the point where stackTwo has just been instantiated, i.e. right after the following statement in the driver.

```
Stack stackTwo (stackOne);
```

Question 4: Draw a diagram to show the memory allocation at the point right before the program terminates, i.e. right before the following statement in the driver

```
return 0;
```

Bonus 2 points

The `Stack` definition provided uses an array of integers. Copy the code into another folder and modify the `Stack` definition so that it uses an array of `Complex` numbers. Be sure to include the appropriate header file(s).

Save your code and use the terminal commands to compile your program using `g++`.

```
$ g++ Stack.cpp stackDriver.cpp –o stackExec
```

Fix any errors or warnings, recompile, and test the program.

```
$ ./stackExec
```

Examine the output and verify that it is as expected.

Now create a script file to demonstrate that your program compiles and executes without errors:

```
$ script stackComplexIN.script
```

where *`IN`* represents both / all partners' initials.

Compile the program again, and run it to demonstrate that it compiles without errors and works correctly.

Exit the script:

```
$ exit
```

Describe the modifications you made to the `Stack` class definition along with a rationale.

Deliverables:

- Clearly labeled and stapled together:
  - This sheet labeled with names of all group members.
  - Neatly typed or written responses to the questions above.
- Zipped file submitted in Canvas under "Exercise – Dynamic Arrays" with
  - Script showing successful compilation and execution of the program.
  - Source code of the modified program (all files).
- Zipped file for **Bonus** submitted in Canvas under "Exercise – Dynamic Arrays" with
  - Script showing successful compilation and execution of the program.
  - Source code of the modified program (all files).