# COMP2710: Homework 6

Points Possible: 20
**Note: You do not need to submit hard copies.**

***Goals:***

- To learn how to define and implement a class
- To learn how to use strings (Note: this topic was covered in Homework 5)
- To learn creating multiple versions via conditional compilation (Note: this topic was covered in Homework 5)
- To use dynamic array
- To learn basic operator overloading
- To perform unit testing

***Description***

Create a class named `Doctor` that has three member variables:

- name – A string that stores the name of the doctor
- numPatients – An integer that tracks how many patients the doctor must treat
- patientList – A dynamic array of strings used to store the patient names

Write appropriate constructor(s), mutator, and accessor methods for the class along with the following:

- A method that inputs all values from the user, including the list of patient names. Note that this method has to support input for an arbitrary number of patient.
- A method that outputs the name and list of all patients.
- A method that resets the number of patient to 0 and the patientList to an empty list.
- An overloaded assignment operator that correctly makes a new copy of the list of patients.
- A destructor that releases all memory that has been allocated.

Write a main method that tests (i.e., unit testing) all of your functions.

***Requirements:***

1. (1 point) Use comments to provide a heading at the top of your code containing your name, Auburn Userid, filename, and how to compile your code. Also describe any help or sources that you used (as per the syllabus).
2. (1 point) Your source code file should be named as "<username>_hw6.cpp" (for example, mine would read "xzq0001_hw6.cpp").

3. (1 point) Your program must use dynamic array
4. (1 point) Your program must use strings
5. (1 point) A method that inputs all values from the user
6. (1 point) A method that outputs the name and list of all patients.
7. (1 points) A method that resets the number of patient to 0 and the patientList to an empty list.
8. (2 points) An overloaded assignment operator
9. (1 point) A destructor
10. (3 points) Correctly implement the main function (i.e., unit testing).
11. (2 points) Creating two versions using conditional compilation.
12. (1 point) You must limit the number of global variables and data
13. (3 points) Usability of your program (e.g., user interface)
14. (1 point) Readability of your source code.

Note: You will lose **at least 2 points (and up to 10 points)** if there are compilation errors or warning messages when the TA compiles your source code. You will lose points if you: do not use the specific program file name, or do not have a comment on each function in your program you hand in.

***How to Create Two Versions?***

You can use the preprocessor directive #ifdef to create and maintain two versions (i.e., a debugging version and a product version) in your program. If you have the sequence

```
#ifdef UNIT_TESTING
add your unit testing code here
#else
add your code for the product version here
#endif
```

in your program, the code that is compiled depends on whether a preprocessor macro by that name is defined or not. For example, if there has been a "#define UNIT_TESTING" macro line), then `` `add your unit testing code here' `` is compiled and `` `add your code for the product version here' `` is ignored. If the macro is not defined, `` `add your code for the product version here' `` is compiled and `` `add your unit testing code here' `` is ignored.

These macros look a lot like if statements, but macros behave completely differently. More specifically, an if statement decides which statements of your program must be executed at run time; #ifdef controls which lines of code in your program are actually compiled.

***Unit Testing:***

Unit testing is a way of determining if an individual function or class works. You need to isolate a single function or class and test only that function or class. For each function in this homework, you need to check normal cases and boundary cases.

Examples for tested values:

- string – empty string, medium length, very long
- Array – empty array, first element, last element
- Int – zero, mid-value, high-value

You must implement a unit test driver for each function implemented in your program. You may need to use assert() to develop your unit test drivers if tested results are predictable.

***Programming Environment:***

Write a short program in C++. Compile and run it using the g++ compiler on a Linux box (either in Shop 3, computer labs in Shelby, your home Linux machine, a Linux box on a virtual machine, or using an emulator like Cygwin).

***Deliverables:***

- Submit your source code file named as "<username>_hw6.cpp" through the Canvas system.

***Late Submission Penalty:***

- Twenty percent (20%) penalty per day for late submission. For example, an assignment submitted after the deadline but up to 1 day (24 hours) late can achieve a maximum of 80% of points allocated for the assignment. An assignment submitted after the deadline but up to 2 days (48 hours) late can achieve a maximum of 60% of points allocated for the assignment.
- Assignment submitted more than 3 days (72 hours) after the deadline will not be graded.

***Rebuttal period:***

- You will be given a period of 72 hours to read and respond to the comments and grades of your homework or project assignment. The TA may use this opportunity to address any concern and question you have. The TA also may ask for additional information from you regarding your homework or project.