Dr. Emily Hill
CMPT 184

TESTING & DEBUGGING HW

Objective

In this homework, you will practice your testing & debugging skills both by answering questions & putting these skills to work using Eclipse.

Steps:

- Download the SortingB project from BB & import it into Eclipse
  Import the SortingB project by opening Eclipse, selecting File > Import and then General > Existing Projects into Workspace. Hit "Next", then "Select archive file," browse to SortingB.zip, and hit "Finish".
- Inside you will find 2 word documents -- Testing & Debugging
- Answer the Testing questions by editing the document, and add the appropriate code as described.
- Answer the Debugging questions by editing the document, and edit the code as appropriate.
- Export your completed SortingB project as an archive file named netid_testingHW.zip -- make sure it contains your latest versions of your two doc files with all your answers!
  You can create your zip file within Eclipse by selecting your project, right-clicking and selecting Export, then selecting "General > Archive File".
- Upload your project zip file to Blackboard. Because both the Testing & Debugging Documents are in your project, you will only need to upload **one** file. If in doubt, also upload the word docs.

Tips:

- Create your test cases for the sort by thinking about the different types of inputs:
  - types of numbers: positive, negative, 0
  - array sizes: 0, 1, many
  - types of array orders: unsorted, sorted, reverse sorted
- Some students write a single test case combining positive numbers, negative numbers, and 0 as a test case and calling their testing done. Such a test case does a good job combining types of numbers in an unsorted array with many elements, but doesn't vary the array size or order of elements. In addition, using a single test case to test types of numbers makes it difficult to find bugs. If this test case fails (and we find a bug), how do we know what caused it? Was it the positive numbers? The negative numbers? The zero? When testing, we need to try combinations of inputs to help us isolate where the bugs are. Just varying the type of input alone (+, -, 0) could result in as many as 7 test cases.
- **Hints for Q5:** When am I done testing? How many test cases (i.e., test methods) do I need? Recall in class we discussed creating test cases using as many combinations of input classes as possible. To completely test a simple method that calculates the area of a rectangle, we may use as many as 20 test cases. Testing a method that sorts an array of numbers is much more complex, and could require even more test cases (I leave that decision up to you).