Homework 1, CS161, Spring 2012, Due Friday, Jan. 27, 5:00 p.m.

The last thing you should do before you turn your program in is to run it from the command line on the department Linux machines. You do this by running "javac Assign1.java" and then "java Assign1". Test it carefully on the Linux machines, because moving a program from Eclipse to the Linux machines can change the behavior of a program, especially if you have bugs that you haven't noticed in Eclipse.

- We have created a program, `Assign1.java`, to get you started. Compile and run the program, play with it, and look over the code.

  The first thing the program does is read two sequences of integers, allocate two arrays to hold them, and store references to the arrays in `Array1` and `Array 2`. You can use a text editor to create input files that have the specified format. Create ones with sequences that are appropriate for testing your methods.

  Selecting menu options 3 and 4 print out the contents of these two arrays using `printArr()`. Find this method's definition and the places where it is called. You should write variants of this method that accomplish the same task using a `for` loop and a `while` loop. Replace the calls generated by options 3 and 4 to test them.

- Menu options 1 and 2 allow you to change the contents of `Array1` and `Array2`. Insert code to the `if` clause and the `else if` clause to do this. Find the place where these arrays are first read in in the program, and use the code that appears there as a guide. Test whether it works using menu options 3 and 4.

- The rest of the assignment consists of filling in the bodies of incomplete methods that we have inserted above the definition of `main`. For each, there is a specification of what the method is supposed to do, and a menu option that you can use to test your solution.

Preconditions and postconditions

With the exception of problems that involve taking inputs from the user, the way every problem is defined in computer science is with *preconditions* and *postconditions*.

- The preconditions tell what conditions must apply before the method is run.

- The postconditions tell what conditions apply after it finishes, *assuming the preconditions have been met.*

This is similar to a contract with the user of your method; it specifies what is expected of them and what is expected of you if they meet their obligations. You get full credit even if there is a loophole in the contract. For example, if they fail to meet their preconditions, you can do anything. You can crash the program, and blame it on them. If they don't like it, they should have relaxed their preconditions to allow bad inputs, and included a postcondition that specifies error handling.

This may seem like an irresponsible attitude, but there are situations where checking whether preconditions are met would be highly inappropriate, since it can ruin the efficiency advantages of an algorithm. Failure to adhere to this discipline leads to misunderstandings between programmers.

Most importantly, there is a time and place to think about what error conditions should be checked, and it's when you define the preconditions and postconditions, *not while you're writing your program.* This is hard to get used to, but once you do, it keeps your mind uncluttered when you write code. When you're writing a method, you have only to think about the precondition and postcondition, and forget all about the motivations for them, such as how the method is used by the larger program. When you're calling a method, you have only to think about the precondition and postcondition, and forget all about the computations that go on inside the method.

This is the key behind the powerful technique of *encapsulation*, which was one motivation for the development of object-oriented programming.

- **Definition:** *An algorithm is* **correct** *if it always meets the postconditions whenever the user meets the preconditions.*

We will never mark you off for a method that implements a correct algorithm, even if we messed up our problem definition. That is the good news. The bad news is that if there exist even one input that meets the precondition and causes algorithm to fail the postcondition, your method is incorrect.

Rules to observe on programming assignments

1. We can't give credit if your program won't compile.
2. Make sure your algorithms satisfy the postconditions whenever the preconditions are met. If the preconditions are not met, you are not responsible for what happens.
3. If you couldn't get a method running reliably, delete the body but leave the stub that you started out with. You will get some credit for this. Part of being a computer scientist is figuring out whether your code is correct. Incorrect code does a lot more economic damage than missed deadlines do. Test your code thoroughly, and don't take chances. Above all, don't take a chance that your program will crash when we try to test it.
4. Every time you declare a new variable, follow it with a comment that tells in an informal conceptual way what the variable is for or what its contents "mean". An exception to this rule is a loop index variable; its meaning is usually obvious. An example in your `arrayMax` method might be the following:

```
int maxSoFar = A[0];  // maximum element seen so far in A[]
```

5. Use appropriate indentation to make the logical structure of your code clear. Follow a standard convention, such as the one used by our textbook.

Submitting assignments

Programming assignments will be submitted using RamCT. You can submit the assignment by selecting Assignments in the Course Menu and selecting the appropriate assignment within the Assignments section. Once you have selected the correct assignment, you should attach all files indicated by the directions of the assignment and submit the assignment for grading. After you have submitted the assignment, be sure to verify that the assignment was submitted correctly by clicking on the submitted tab under Assignments and verifying that the files that were submitted are correct. Attaching incorrect files is one of the most common problems - it is the responsibility of the student to ensure that the correct files were submitted for the assignment.

Once an assignment has been submitted, you can "unsubmit" the assignment by locating the assignment under the submitted tab, clicking on the pull-down menu associated with the assignment and selecting the "Take back assignment to Inbox". This will allow you to modify the attached files and resubmit the assignment. BE SURE to submit the modified assignment when you are finished.