

II. B) More Comparing Strings (5.2.3 p.193)

1a) Open **AP1ch05lab02iib.java**

b) Run and write the last 3 lines of output: computer word is _____

input word is _____

The 2 words are _____

2a) Change **if (computerWord.equals(inputWord))** to
if (computerWord.equalsIgnoreCase(inputWord))

b) Run and write the last 3 lines of output: computer word is _____

input word is _____

The 2 words are _____

C) More Comparing Strings (5.2.3 p.193)

1a) Open **AP1ch05lab02iic.java**

b) Run using as input: cat and write the last 3 lines of output: computer word is _____

input word is _____

The computer word _____ comes after the input word _____

c) Run using as input: pig and write the last 3 lines of output: computer word is _____

input word is _____

The computer word _____ comes before the input word _____

d) Run using as input: Pig and write the last 3 lines of output: computer word is _____

input word is _____

The computer word _____ comes after the input word _____

Summary: The `compareTo()` method returns a value of -1 if the string comes earlier in the alphabet, and it returns 1 if later in the alphabet and 0 if they are equal strings. Capital letters, though, come before lower case ones when comparing the order!

Turn in this sheet to be graded!

I. A) Comparing Floating-Point Number (5.2.2 p.192)

1a) Open **AP1ch05lab02a.java**

b) Run and write the output:

The square root of 3 squared _____

Note: This is another example of roundoff error. When comparing values that are floating-point numbers you need to be careful of this error.

2a) Change **if (y == 3)** to **if (Math.abs(3 - y) < 0.000001)**

b) Run and write the output:

The square root of 3 squared _____

Summary: To compare floating-point numbers use the above code using the difference inside of an absolute value and compare that to a small value like 0.0000001 (which is predetermined ahead of time how small of a value to use).

II. A) Comparing Strings (5.2.3 p.193)

1a) Open **AP1ch05lab02iia.java**

b) Run and write the last 3 lines of output:

computer word is _____

input word is _____

The 2 words are _____

2a) Change **if (inputWord == computerWord)** to **if (inputWord.equals(computerWord))**

b) Run and write the last 3 lines of output:

computer word is _____

input word is _____

The 2 words are _____

3a) Change **if (inputWord.equals(computerWord))** to **if (computerWord.equals(inputWord))** and then Run

b) Did you get the same result as #2b)? (yes/no) _____

Summary: When strings are stored they are stored just like objects as a reference number to a memory location. `inputWord` above has a number associated with it that refers (points) to a memory location that has "hi" stored in it and `computerWord` has a different number associated with it that refers (points) to a different memory location that has another "hi" stored in it. When `==` is used to compare strings the computer is really comparing the memory numbers which are not the same. To compare the actual contents ("hi") of the memory locations you must use the method `equals()` which is a non-static method found in the String class.

OVER