AP Computer Science Name _____ Per ____

Semester 2 Final Exam Study Guide

In General:

- KNOW THE APCS QUICK REFERENCE GUIDE!
- Take the practice exams in your AP Test Prep Guide or the Barron's AP Computer Science study guide.
- Answer multiple choice questions in groups of 5 in your booklet and then bubble in your choices in groups of 5 on your answer document. You will save time.
- Scan the question and read the answer selections BEFORE working through any code. The answer choices will give you clues as to what to look for.
- When a range of numbers is expressed in interval notation `[0,5)`, the "`[`" means to include the value, "`)`" means to include all real numbers right up to but NOT including the value. `[0,5)` would include all numbers from 0, including 0, to 4.999999…, NOT including 5.

logic

- Understand the meaning of "short-circuit evaluation". The subexpressions in a compound boolean expression are evaluated from left to right, and evaluations stops as soon as the value of the entire expression is known.

```
A && B                                    A || B
```

- If an algorithm checks for `(x > 5)`, check the algorithm with all possibilities, when `x` is greater than 5, when `x` is equal to 5 and when `x` is less than 5. (Plug in values 6, 5 and 4 for `x`).

- For problems like:

`!(a > b) || (a == b)` create a t-chart with numbers where `a > b`, `a = b` and `a < b`

a	b
2	1
1	1
1	2

- Make a t-chart for testing `boolean` values in questions like:

```
!(x && y) || (!x &&!y)
```

x	y
T	T
T	F
F	T
F	F

String class

- Know how to use the substring method.

```
String str = "Hello";
```

- What would be returned with the following calls?

```
str.substring(1);                 str.substring(2,4);
str.substring(2,5);               str.substring(3,6);
```

- Use the `length()` method of the `String` class to find the number of characters in a string.

`str.length()` returns?

- Use the `indexOf` method to find the index of the first occurrence of a string in another string.

`str.indexOf("ll")` returns? `str.indexOf("HEL")` returns? `str.indexOf("He")` returns?

Arrays `[]`

- Arrays are objects. They contain an instance variable `length`, which contains the number of cells in the array.
- Use a `for` loop to examine the cells in an array.

Write code to loop through an array of integers, putting the even numbered cells into the odd number cells (0 is even). For example, the array {3, 4, 1, 2, 9, 7}, would change to {3, 3, 1, 1, 9, 9}

ArrayList class

- Know that the `ArrayList` class implements the `List` interface. The method headers are defined in the `List` interface.
- `ArrayList` reference variables can have a type of `List`, but objects cannot be instantiated from the `List` interface. For example:
  OK – `List<String> names = new ArrayList<String>();`
  NOT OK – `List<String> names = new List<String>();`
- Know all of the listed methods for the `ArrayList` class (they are listed under the `List` interface in the *Quick Reference Guide*).
- Narrow the answer choices by eliminating answers with `[]`. `ArrayList` never uses `[]` to access a cell.
- To determine the size of an `ArrayList`, use the `size()` method.
- `ArrayList` cannot contain primitive types. Use wrapper classes `Integer`, `Double` to store `int` and `double` types.

Write code to print out the contents of a `String ArrayList` using an enhanced `for` loop.

double data types

- Do not rely on == to test for equality or `!=` to test for inequality on `double` values. There is always a chance that a round-off error can occur.
- `double` data types may also be referred to as "floating point" values (contains a decimal point)
. `4.56, 5.01, .001, 15.00` are all `double` or floating point values.
- `Math.random()` returns a `double` between 0 and .99. Know how to use this method to generate random numbers between two values.

Generate a number between 0 – 10.

Generate a number between 1 – 10.

% mod operator (remainder)

- Know how to use the `%` operator. `x % y` is read "what is the reminder when `x` is divided by `y`"

1 % 3 = 4 % 2 = 3 % 10 = 17 % 3 =

- Use the `%` operator to examine the digits of an integer. For example, to look at each digit in the number 3456 use the following algorithm:

```
n = 3456;
while (n > 0)
{
      int digit = n % 10;
      n = n / 10;
}
```

Know what is in the APCS Java subset:

- `boolean`, `int`, and `double` primitive data types. `(int)` and `(double)` casts. **Other primitive data types, including `char`, are not in the subset and should be avoided on the exam.**

- Assignment (=), arithmetic (+, -, *, /, %), increment/decrement (++, --), compound assignment (+=, -=, *=, /=, %=), relational (<, >, <=, >=, ==, !=), and logical (&&, ||, !) operators. **Use only the postfix form of `++` and `--` (`x++` or `x--`), and do not use them in expressions**.

- `+` and `+=` operators for concatenating strings. `String`'s `compareTo`, `equals`, `length`, `substring`, and `indexOf(String s)` methods. `\n`, `\\`, and `\"` escape sequences in literal strings.

- `System.out.print` and `System.out.println`.

- One- and two-dimensional arrays, `array.length`, arrays of objects, initialized arrays such as `int[] x = {1,2,3};`

- `if-else`, `for`, including the "for each" form, `for(`*`type`* `x : values)...`, `while`, `return`.

Classes

- Constructors, the `new` operator, `public` and `private` methods, `static` methods, `static` variables and `static final` variables (constants), overloaded methods, `null`. **All instance variables are `private`**. Default initialization rules are not in the subset and won't come up on the exam.

- Inheritance, interfaces and abstract classes, `extends`, `implements`. Calling a superclass's constructor from a subclass (as in `super(...)`). Calling a superclass's method from a subclass (as in `super.someMethod(...)`).

- Passing `this` object to a method (as in `otherObject.someMethod(this)`).

- `NullPointerException`, `ArrayIndexOutOfBoundsException`, `ArithmeticException`, `IllegalArgumentException`, `ClassCastException`.

Library classes, methods, and constants

`String`: `length()`, `substring(...)`, `indexOf(String s)`
`Integer`: `Integer(int x)`, `intValue()`; `Integer.MIN_VALUE` and `Integer.MAX_VALUE`.
`Double`: `Double(double x)`, `doubleValue()`
`Math`: `abs(int x)`, `abs(double x)`, `pow(double base, double exp)`, `sqrt(double x)`, `random()`

- Also understand `toString` methods for all objects, the `equals` and `compareTo` methods for `String`, `Integer`, and `Double`, and the `Comparable<`*`T`*`>` interface.

- The `List<`*`E`*`>` interface and the `ArrayList<`*`E`*`>` class

Exam Taking Tips

Some things are obvious:

- If you took the time to read a multiple-choice question and all the answer choices but decided to skip it, take an extra ten seconds and guess. Most likely you have eliminated one or two wrong answers even without noticing.

- If a common paragraph refers to a group of questions and you took the time to read it, try each question in the group.

- Do read the question before jumping to the code included in the question.

There are a few important things to know about answering free-response questions.

Remember that all free-response questions have equal weight. Don't assume that the first question is the easiest and the last is the hardest.

In a nutshell: be neat, straightforward, and professional; keep your exam reader in mind; don't show off.

More specifically:

1. Stay within the AP Java subset.

2. Remember that the elegance of your code does not count. More often than not, a brute-force approach is the best. You may waste a lot of time writing tricky, non-standard code and trick yourself in the process or mislead your exam reader who, after all, is only human. No one will test your code on a computer.

3. Superior efficiency of your code does not count, unless the desired performance of the solution is specifically stated in the question.

4. Remember that Parts (b) and (c) of a question are graded independently from the previous parts, and may actually be easier: Part (a) may ask you to write a method, while Part (b) or Part (c) may simply ask you to use it. It is common for method(s) specified in Part (a) to be called in subsequent parts. Do so, even if your Part (a) is incorrect or left blank. Do not re-implement code from earlier parts in later parts — you will waste valuable time and may lose points for doing so.

5. Bits of "good thinking" count. You may not know the whole solution, but if you have read and understood the question, go ahead and write fragments of code that may earn you partial credit points. But don't spend too much time improvising incorrect code.

6. Don't waste your time erasing large portions of work. Instead, cross out your work with one neat line, but only after you have something better to replace it with. Do not cross out a solution if you have no time to redo it, even if you think it is wrong. You won't be penalized for incorrect code and may get partial credit for it. Exam readers are instructed not to read any code that you have crossed out. But if you wrote two solutions, be sure to cross one out: otherwise only the first one on the page will be graded.

7. Read the comment above the method header quickly — it usually restates the task in a more formal way and sometimes gives hints. Assume that all preconditions are satisfied — don't add unnecessary checks to your code.

8. One common mistake is to forget a `return` statement in a non-`void` method. Make sure the returned value matches the specified type.

9. Do not ignore any hints in the question description. If an algorithm is suggested for a method (as in "you may use the following algorithm"), don't fight it, just do it! If the description says "you may use a helper method," be sure to write and use one: chances are it is much more difficult to come up with a solution without a helper method.

10. Remember that the exam readers grade a vast number of exams in quick succession during a marathon grading session every June. Write as neatly as possible. Space out your code (don't save paper).

11. Always indent your code properly. This helps you and your exam reader. If you miss a brace but your code is properly indented, the reader (as opposed to a Java compiler) may accept it as correct. Similarly, if you put each statement on a separate line, a forgotten semicolon may not be held against you.

12. Follow the Java naming style: the names of all methods, variables, and parameters start with a lowercase letter. Use meaningful, but not too verbose, names for variables. `count` may be better than `a`; `sum` may be better than `temp`; `row`, `col` may be better than `i`, `j`. But `k` is better than `loopControlVariable`. If the question contains examples of code with names, use the same names when appropriate.

13. Don't bother with comments; they do not count and you will lose valuable time. Occasionally you can put a very brief comment that indicates your intentions for the fragment of code that follows. For example:
```
// Find the first empty seat:
...
...
```

14. Don't worry about `imports` — assume that all the necessary library classes are imported.

15. Code strictly according to the specifications and preconditions. Avoid extraneous "bells and whistles" — you will lose points. Never add `System.out.print/println` in solutions unless specifically asked to do so.

16. Use recursion when appropriate: if specifically requested or especially tempting.

17. Don't try to catch the exam authors on ambiguities: there will be no one to hear your case, and you'll waste your time. Instead, try to grasp quickly what was meant and write your answer.

18. Don't quit until the time is up. Use all the time you have and keep trying. The test will be over before you know it.