

Name: _____

Date: _____ Per: _____

AP Computer Science, Mr. Ferraro

APCS Semester #1 Final Exam Practice Problems

SOLUTIONS (for problems 1–25)

The problems here are to get you thinking about topics we've visited thus far in preparation for the semester final. Note that the problems here are not all-inclusive and that you may need to review other topic areas in order to prepare yourself adequately!

You may work with at most one other person. Whenever possible, **verify your answers using a Java compiler.**

1. Is it ever valid Java syntax to write an `if()` statement without curly braces? If so, write a brief example.

Yes. For example,

```
if ( true )
    System.out.println("hi!");
```

2. May `for()` statements ever be written without curly braces? Again, provide an example if that omission is allowed.

Yes. For example,

```
for( int i = 0 ; i < 10 ; i++ )
    System.out.println(i);
```

3. Will the code below run as is? If not, suggest changes.

```
int b = 18;
for ( int a = 1 ; a < b ; a++ )
    if ( b % a == 0 )
        System.out.println( a + " is a factor of " + b );
```

Yes, it will run. The `if()` statement is just one statement (spanning two lines). Since the `for()` statement has a single statement in its execution block, curly braces may be omitted around the `if()`.

4. May a `for()` loop be run without a code block, as in the example below?

```
int e, f = 65;
for ( e = 0 ; e*e <= f ; e++ );
```

Yes. The `for()` loop is equivalent to this `while()` loop:

```
e = 0;
while( e*e <= f ) {
    e++;
}
```

5. If the goal of the code snippet in question 4 was to find the greatest integer whose square is $\leq f$, write a line of code to follow the given snippet that will print that integer. For example, if $f = 17$, then you would print 4, since 4 is the largest integer such that $4^2 \leq 17$.

Remember that `e` will be incremented to 5 before the `for()` loop's condition fails:

```
System.out.println( e-1 );
```

For questions 6-8, refer to the code segment below.

```
int i, j = 12;
for ( i = 0 ; i < j ; i+=2 );
```

6. How many iterations are performed? **6**
7. What is the final value of `i`? **12**
8. If `j` were instead initialized to 13,
- (a) how many iterations? **7**
 - (b) final value of `i` = **14**
9. Write nested `for()` loops that would result in the printing of

0,0 ; 0,1 ; 0,2 ; 1,0 ; 1,1 ; 1,2 ; 2,0 ; 2,1 ; 2,2 ; 3,0 ; 3,1 ; 3,2 ;

The first number in each pair (`i`) goes from 0→3, while the second number (`j`) goes from 0→2.

```
for( int i = 0 ; i < 4 ; i++ ) {
    for( int j = 0 ; j < 3 ; j++ ) {
        System.out.println(i + "," + j + " ; ");
    }
}
```

10. *Variable Declaration vs. Initialization:* When a variable is declared, a *variable declaration* may be written as follows:

```
//for a primitive
int q;

//for an object
Square sq1;
```

A variable, which has been declared, may now (or later) be *initialized*. For example,

```
q = 1;

sq1 = new Square();
```

Using the terms *declaration* and *initialization*, explain what happens when this code is run:

```
double trouble = -2.9;
```

A variable named **trouble** is **declared** as data type **double**. **trouble** is **initialized** to **-2.9**.

11. Write an expression that evaluates to the remainder when 186 is divided 4.

```
186 % 4
```

12. In the snippet below, is the use of **else** necessary? If not, rewrite the snippet without **else** in a manner that preserves the original behavior.

```
if ( a == 0 ) {
    return true;
} else {
    return false;
}
```

No, the **else** is not necessary:

```
if ( a == 0 ) {
    return true;
}
return false;
```

13. Rewrite the `return` line in `getWinPercentage()` so that it returns an `int` representing the win percentage. *Note: You need not consider the case of a starting scoreboard in which `wins = losses = aborts = 0`.*

```
int wins = 3, losses = 2, aborts = 0;

public int getWinPercentage() {
    return wins / wins + losses + aborts * 100;
}
```

Don't make the mistake of casting into an `int` before multiplying by 100!

```
return (int)((double)wins / (wins + losses + aborts) * 100);

//better versions: round off value; Math.round() returns long, so cast
return (int)((double)wins / (wins + losses + aborts) * 100 + .5);
return (int)Math.round((double)wins / (wins + losses + aborts) * 100);
```

14. For each `System.out.println()`, provide its output.

- (a) `System.out.println("My lucky number is " + 3);`
`My lucky number is 3`
- (b) `System.out.println(3 + " is my lucky number");`
`3 is my lucky number`
- (c) `System.out.println(-5 + 8 + " is my lucky number");`
`3 is my lucky number`
- (d) `System.out.println("My lucky number is " + -5 + 8);`
`My lucky number is -58`

15. List as many roles of an operating system as you can. (Don't include extra features like a GUI calculator and a media player; just the core features!)

- act as middleman between other programs and the hardware
- enable multitasking of programs, allowing multiple programs to run simultaneously
- brokering hardware resources, enabling communication between programs and hardware
- abstracting away hardware differences so programs can run on a range of hardware without any (or substantial) changes
- ensuring that other programs share resources
- there's plenty more: managing file systems, user account management and security, etc.

16. What is the name of the exception that lets you know when you try to access an element in an `ArrayList` that does not exist? If you don't remember, create an `ArrayList` and then try to `get()` an index that is negative or \geq `size()`.

`IndexOutOfBoundsException`

17. Define the following:

- (a) *constructor*: a method that returns a reference to a newly-created object; method may optionally set initial field values and perform other tasks.
- (b) *accessor method*: also referred to as a *getter*, returns the value of a field that is typically declared `private`; takes no arguments, returns a value or object reference.
- (c) *mutator method*: also referred to as a *setter*, takes one or more arguments and changes the value of a field; usually a `void` method.

18. Why is the keyword `private` used for fields? What is the practice called?

`private` is used to ensure that other classes aren't able to make unauthorized changes to the internal representations inside objects of the current class. The practice is called *data hiding* and *encapsulation*.

19. Methods take `arguments` or `parameters`. (*There are two correct, one-word responses.*)

20. The use of `void` indicates what?

`void` indicates that a method returns nothing.

21. What's the difference between *binary search* and *sequential search*? When is each one good to use?

Binary search eliminates roughly half of the items in a set being examined with each step, while sequential search eliminates only one element per step. If the items are in some kind of order, binary search is typically more efficient.

22. A boolean expression is one that evaluates to `true` or `false`.

23. To this boolean expression —

```
a < b + 2 || a >= 6 && a == 0
```

— add parentheses to show the order in which Java will evaluate the operators.

```
(a < (b + 2)) || ((a >= 6) && (a == 0))
```

24. Write an expression using `Math.random()` that will evaluate to a random integer between 1 and 50.

```
(int)(50 * Math.random()) + 1
```

25. This problem requires the `Fraction` class from Litvin Ch. 9. Test out your answers after writing them below to verify correctness.

- (a) Write a statement that creates `Fraction f5` whose value is $\frac{3}{9}$:

```
Fraction f5 = new Fraction(3,9);
```

- (b) Write a statement that changes `f5`, adding $\frac{1}{2}$ to its value:

```
f5 = f5.add( new Fraction(1,2) );
```

- (c) Predict the output of `System.out.println(f5 + " " + f5.getValue())`, assuming your answers above are correct:

```
5/6 .833333333333
```

Programming Exercises

26. Write a program that prints the number of years it will take for a \$150 deposit to reach \$800 if the annually compounded interest rate is 8%.

27. Write a method called `printStars()` that takes an argument, `n`, and prints out a triangle with `n` rows, shaped like the one below:

```
*
**
***
****
```

28. Write `printStarsReverse()`, which does the same as `printStars()`, but prints a triangle like this:

```
****
***
**
*
```

29. Write a method called `sumOneToN()` that computes — and prints — the sum of all integers from 1 to `n`. Your method must take `n` as an argument. For example, if your method is called with 5 as the argument, it should print the result of $1 + 2 + 3 + 4 + 5$, or 15.

30. Make a method `sumOneToN2()` that functions in exactly the same manner as `sumOneToN()`, but does so in the opposite manner, meaning that this version counts down if your original counts up, or vice versa.

31. Write `sumOneToNRec()` that functions in the same manner as the prior versions, but does so via recursion.