Name: _____

Introduction to Computer Science I
Spring 2018
Midterm

This is a 50-minute, closed-notes exam. **Please put away all notes, phones, laptops, etc.** There are four problems, each of which is worth 25 points. Good luck!

Problem	**Score**
1	/25
2	/25
3	/25
4	/25

If you are unsure of what a problem is asking you do to, write down the question that you would like to ask me to help you clarify the problem, and write what you think my answer would be. If you have misinterpreted the problem, I will take this into account when grading.

You do not need to show any work beyond your final answer, but you are welcome to do so if you feel it will help me understand your thought process.

1. None of the following bits of code are correct: some have compiler errors, others have runtime errors, and others will run but do not produce the desired behavior. For each:

   - Briefly (in a few words or a sentence) **explain what the mistake is**.
   - Make a *small* correction to **fix the error**. Your corrections should *not* change the type of any variables.

   (a) Desired behavior: program prints `5/10 = 0.5`. (Do not simply change the print statement to say `System.out.println("5/10 = 0.5")`.)

   ```
   int x = 5;
   int y = 10;
   System.out.println(x + "/" + y + "=" + (x/y) );
   ```

   Solution: The problem is that the division at the end, `x/y`, is int division, so `5/10` will evaluate to 0. One way to fix this is to cast `x` or `y` to a double: `((double)x/y)`. Note that `(double)(x/y)` will not work, because this will compute `x/y` using int division and then cast the result to a double.

   (b) Desired behavior: program prints "big" if `i` is greater than 10, and "small" otherwise.

   ```
   int i;
   if(i > 10) {
      System.out.println("big");
   }
   else {
      System.out.println("small");
   }
   ```

   Solution: The problem is that the variable `i` is never initialized (we haven't stored a value in `i`), so when we go to check the condition `i > 10` we will get an error. One way to fix this is to change the first line of code to say `int i = 5;` (or whatever value we want `i` to have).

(c) Desired behavior: at the end of the program, the variable `x` contains a number between 2 and 10 (inclusive). (Assume that some earlier part of the program contains the code needed to use the keyboard.)

```
System.out.println("Enter an integer between 2 and 10");
int x = keyboard.nextInt();
while( !(2 <= x <= 10) ) {
   System.out.println("Enter an integer between 2 and 10");
   x = keyboard.nextInt();
}
```

Solution: The problem here is that we can't chain together inequalities in this way. Java will evaluate `2 <= x <= 10` from left to right, so we first evaluate `2 <= x`. This is either `true` or `false`. Then we try to evaluate `true <= 10`, which makes no sense. We can fix this by using the and operator to combine two inequalities: `while( !(2 <= x && x <= 10) )`.

(d) Desired behavior: at the end of the program, the variable `i` contains the value 11. (Do not simply set `i = 11`, and do not change the values stored in `p` and `q`.)

```
int p = 15;
double q = 4.5;
int i = p - q;
```

Solution: This is a type error: `p` is an int and `q` is a double, so `p - q` is a double and thus can't be stored in `i`, which is an int. Fixing this is a little tricky! If we cast `p - q` to an int, we will end up storing 10 instead of 11 (because $15 - 4.5 = 10.5$, which will be truncated to 10). Instead, one way to fix this is to cast `q` to an int: `int i = p - (int)q;`.

(e) Desired behavior: program prints the sum of all integers from 1 to `x`. (Assume that some earlier part of the program contains the code needed to use the keyboard.)

```
int x = keyboard.nextInt();
for(int i = 0; i < x; i++) {
   int sum = sum + i;
}
System.out.println(sum);
```

Solution: This is a scope problem. The variable `sum` is declared inside the body of the for loop, so `sum` no longer exists when we get to the print statement. We can fix this by adding the line `int sum = 0;` at the beginning of this piece of code, and then remove the word `int` on the line in the for loop body.

2. Consider the following piece of code.

```
int i = 8;
int j = 5;

if( !(i % 3 == 2) || i + j < 10 ) {
   i = i - j;
   System.out.println("Message 1:" + (i + j) );
}
else if( i/j == 1 && j - i <= 0 ) {
   i = i + j;
   System.out.println("Message 2:" + (i + j) );
}
else {
   j = 3 * i;
   System.out.println("Message 3:" + (i + j) );
}
System.out.println("Last message:" + i + j);
```

What is the output of this program? That is, what prints when it runs?

Solution: This program will print:

```
Message 2:18
Last message:135
```

We first check the condition in the `if` statement. `i` is 8, so `i % 3` is equal to 2, so `!(i % 3 == 2)` evaluates to false. On the other side of the or, `i + j` is 13, which is not less than 10, so the second part of the or also evaluates to false. The entire if condition thus is false, and we jump down to the else if.

We now check the condition in the `else if`. Using int division, `i/j` evaluates to 1, so the first part of our condition `i/j == 1` is true. The second part of our condition says `j - i <= 0`, which is also true because $5 - 8 = -3$. So our condition is now `true && true`, which evaluates to true and we move into the curly braces. We set `i` to the value `i + j`, which is 13, and then print out "Message 2: 13."

Because we executed the block of code associated with the `else if` condition, we'll skip over the `else` block and move to the code below the last curly brace. At this point `i` has value 13 and `j` has value 5. The last line will print "Last message:135." We didn't add the values of `i` and `j` this time because we evaluate the expression from right to left. The first thing that happens is we concatenate the string "Last message:" with `i`, resulting in the string "Last message:13", which then gets concatenated to `j` giving us "Last message:135".

3. Consider the following program:

```
int x = keyboard.nextInt();
boolean done = false;
int y = x - 1;
while(!done && y > 1) {
   if(x % y == 0) {
      done = true;
   }
   y--;
}
System.out.println("The answer is: " + done);
```

(a) What prints when the user enters the number 5?

Solution: The answer is: false

(b) What prints when the user enters the number 4?

Solution: The answer is: true

(c) In general, this program answers a question about the number the user typed in. What is that question? In other words, under what circumstances will the code print `The answer is true`, and under what circumstances will it print `The answer is false`?

The program will print `false` if the number is prime and `true` if the number is composite. It works by starting `x` at the number the user enters and `y` at `x-1`. On each iteration of the while loop, we check if `x` is divisible by `y`. If it is (in which case `x` is composite), we set our `done` flag to `true`. The next time we hit the while condition, the `!done` part will evaluate to false, breaking us out of the loop. If instead `x` wasn't divisible by `y`, we don't set `done` to `true`, decrement `y`, and keep going.

4. The game Fizz-Buzz works as follows: you start counting up from 1, but every time you get to a multiple of 3 you say "fizz" instead of the number, and every time you get to a multiple of 5 you say "buzz" instead of the number. If the number is a multiple of both 3 and 5, like 15, you say "fizz buzz."

   Write a `for` loop to print out the numbers 1 through 100, Fizz-Buzz style. The output of your program should look something like:

   ```
   1
   2
   fizz
   4
   buzz
   fizz
   7
   8
   ```

   And so on, continuing through 100 (which should print as "buzz"). For numbers that are multiples of both 3 and 5, "fizz buzz" should print out on a single line.

   Solution:

   ```
   for(int i = 1; i <= 100; i++) {
      if(i % 3 == 0) {
         System.out.print("fizz ");
      }
      if(i % 5 == 0) {
         System.out.print("buzz");
      }
      if(i % 3 != 0 && i % 5 != 0) {
         System.out.print(i);
      }
      System.out.println();
   }
   ```

   Other solutions also are possible. For example, inside our for loop we could have chained together an if/else if/else block to check a sequence of conditions:

```
if(i % 3 == 0 && i % 5 == 0) {
   System.out.println("fizz buzz");
}
else if(i % 3 == 0) {
   System.out.println("fizz");
}
else if(i % 5 == 0) {
   System.out.println("buzz");
}
else {
   System.out.println(i);
}
```

Or any number of alternatives.