NAME: ________________________________

• Do not turn the page until instructed to do so.

• This booklet contains 10 pages including the cover page.

• This is a closed-book exam. All you need is the exam and a writing utensil. (You may use a calculator if you wish.)

• You have exactly 55 minutes.

• The maximum possible score is 55.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>55</td>
<td></td>
</tr>
</tbody>
</table>
1. (10 pts.) TRUE/FALSE.

For each of the following, indicate whether the statement is true or false.
(\textit{Note: I am including some explanations for a few of these; this is just for the purpose of the answer key, and you will not be expected to do the same on the actual exam.})

a. A \textbf{public} instance variable \emph{can be} accessed and altered by any class, including the class in which it is declared.

\textbf{True}: this is just the definition of \textbf{public} access.

b. A \textbf{private} instance variable \emph{can not be} accessed and altered by any class, including the class in which it is declared.

\textbf{False}: while the variable can not be accessed by any \emph{other} class, it can still be accessed and altered by its own (declaring) class.

c. We can make local method variables either \textbf{public} or \textbf{private}, as we choose.

\textbf{False}: global instance variables are \textbf{public} or \textbf{private}; local method variables do not use these access modifiers at all.

d. If we create a \textbf{new} instance of an object in our code, then we can call any method from that class that we like.

\textbf{False}: we can only call the \textbf{public} methods.

e. You can create a global variable and a local method variable in the same class, with the same name, and with the same type.

\textbf{True}: if we do so, the method will use the local version instead of the global.

f. If the boolean condition for a \textbf{while} loop is \textbf{false}, then the loop will never run.

\textbf{True}: the condition is always evaluated before the loop can run, even once.

g. If a loop does not make progress, then it will run infinitely.

\textbf{False}: for example, the loop may \emph{never run at all}.

h. An integer counter variable used in a \textbf{for} loop is \emph{always} local to the loop.

\textbf{False}: we can, if we like, declare the variable \emph{before} the loop header, so it will be accessible in a higher scope.

i. Within a class, methods can use input variables with the same names as input variables in other methods.

\textbf{True}: input variables are local to their methods.
j. A non-void method must _always_ have a `return` statement.

**True:** while the `return` is optional for `void` methods, it is mandatory in this case.
2. (10 pts.) SHORT ANSWER.

a. (3 pts.) How many times will the following loops run (assuming they are in a correct program)?

(1) int num = 0;
    while ( num < 11 ) {
        System.out.println( num );
        num = num + 1;
    }

Answer: 11 times (for num == 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10).

(2) for ( int i = 0; i <= 10; i += 2 ) {
    System.out.println( i );
}

Answer: 6 times (for i == 0, 2, 4, 6, 8, 10).

(3) for ( int j = 0; j < 10; j = j + 3 ) {
    System.out.println( j );
}

Answer: 4 times (for j == 0, 3, 6, 9).

b. (3 pts.) List three things that make up a method signature (i.e., the top line of the method, when you are creating it yourself), not including the name of the method.

(1) Access control (public or private).
(2) Return type (void or some other non-void type).
(3) Input parameters (or at least empty parentheses to indicate no inputs used).

c. (4 pts.) Suppose we have a class, Driver, and in that class we call a method on a Gadget object:

    Gadget g = new Gadget();
    String s = g.make( 10.5, "Test" );

Without knowing what the make() method does, we do know what its method declaration (i.e., its first line) will look like. Write the method declaration.
**Answer:** Here is what the method will look like (note that the input variable names can be different, but everything else about the method is as shown here):

```java
public String make( double num, String s )
```
3. (10 pts.) CODE EVALUATION.

a. Suppose we run the following method, with input "Hello". Write out what will be printed.

```java
private void method1( String sin ) {
    String sout = new String();
    for ( int i = 0; i < sin.length(); i++ ) {
        sout = sin.charAt( i ) + sout;
        System.out.println( sout );
    }
}
```

Answer: the code outputs the following:

H
eH
leH
lleH
olleH

b. Write out what will be printed by the following method on inputs 5 and 10, in that order.

```java
private void method2( int num1, int num2 ) {
    for ( int i = 0; i < num1; i++ ) {
        System.out.print( i + " : " );
        int j = i;
        while ( j < num2 ) {
            System.out.print( j + " " );
            j = j + 2;
        }
        System.out.println( "END" );
    }
}
```

Answer: the code outputs the following:

0: 0 2 4 6 8 END
1: 1 3 5 7 9 END
2: 2 4 6 8 END
3: 3 5 7 9 END
4. (10 pts.) CODING NESTED LOOPS

Fill in the `main()` method in the class below so that when it runs it prints output (using `System.out`) that looks like this:

```
1 2 4 8
2 4 8 16
3 6 12 24
4 8 16 32
5 10 20 40
```

For full points, your code must use a pair of **nested loops**, each of which is actually used to generate the output. (You may use whatever types of loops you choose.)

**Answer:** Here are two solutions. One uses `for` loops, the other uses `while` loops.

```java
class Main {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            int num = i;
            for (int j = 0; j < 4; j++) {
                System.out.print(num + " ");
                num = num * 2;
            }
            System.out.println();
        }
    }
}

class Main {
    public static void main(String[] args) {
        int x = 1;
        while (x <= 5) {
            int num = x;
            int y = 0;
            while (y < 4) {
                System.out.print(num + " ");
                num = num * 2;
                y++;
            }
            System.out.println();
        }
    }
}
```

    System.out.println();
    x++;
    }
    }
    }
    }
5. (15 pts.)  CODE COMPLETION.

On the next page, complete the given Driver class as follows:

a. Write the method removeVowels() so that it works with the code as given:
   i. It will take a String as input.
   ii. It will return a String as output. The output will be identical to the input, but with any
       lower-case vowels (a, e, i, o, u) removed.

b. Write the method longest() so that it works with the code as given:
   i. This method will take two String inputs.
   ii. It will return as output the String that is the longest of the two inputs. (If they are of
       the same length, then it should return the first one input.)

c. Write the method swapChars() so that it works with the code as given:
   i. This method should take in two char inputs and a single String input.
   ii. It should return a String. The output should be identical to the input String, but with
       every occurrence of the first char replaced with the second char.

When complete, the code should produce the following output when run.

Starting string = Pork tacos
String without vowels = Prk tcs
Longest string = Pork tacos
Swapped string = Perk taces
public class Driver {
    public static void main( String[] args ) {
        Driver d = new Driver();
        String s = "Pork tacos";
        String noVowels = d.removeVowels( s );
        String longest = d.getLongest( s, noVowels );
        String swap = d.swapChars( 'o', 'e', s );
        System.out.println( "Starting string = " + s );
        System.out.println( "String without vowels = " + noVowels );
        System.out.println( "Longest string = " + longest );
        System.out.println( "Swapped string = " + swap );
    }

    private String removeVowels( String s ) {
        String out = "";
        for ( int i = 0; i < s.length(); i++ ) {
            char c = s.charAt( i );
            if ( c != 'a' && c != 'e' && c != 'i' && c != 'o' && c != 'u' ) {
                out = out + c;
            }
        }
        return out;
    }

    private String getLongest( String s1, String s2 ) {
        if ( s1.length() >= s2.length() ) {
            return s1;
        } else {
            return s2;
        }
    }

    private String swapChars( char c1, char c2, String s ) {
        String out = new String();
        for ( int i = 0; i < s.length(); i++ ) {
            char c = s.charAt( i );
            if ( c == c1 ) {
                out = out + c2;
            }
        }
        return out;
    }
}
else {
    out = out + c;
}

return out;