Tutorial 5

You will be expected to engage with the tutor and discuss solutions to the problems presented here. The content covered in the tutorials will assist in your understanding of the practical requirements for the checkpoints.

1. Logical Operators

i. What types of values are used with *logical operators* and what type of value is returned?

ii. What is the value of each of the following expressions:

```
a. !true

b. false || true

c. true || false

d. true && (false || true)

e. !true && !false

f. false || false || false || true

g. !(true && (false || !true))
```

iii. In the class `BoolTest` below, the methods `mystery0`, `mystery1`, etc. do not make use of logical operators. Examine the statements in the methods and determine what logical operations they are performing.

One way to approach this is to see what value a method call will return for all possible inputs (note that execution of a `return` statement causes a method to terminate and the value of the expression after the "return" to be returned). The end result will be a truth table. For example, the value of the method call `mystery0(true, true)` is the value returned by the block part of `mystery0` with the formal parameters being assigned the value of the actual parameters:

```
boolean x = true;
boolean y = true;

if (x) return y;
return false;
```

The result is `true`. Thus the the truth table for `mystery0` will look like:

x	y	mystery0(x,y)
true	true	true
true	false	?
false	true	?
false	false	?

Fill in the missing values for the `mystery0` table and see if you can find a truth table in the notes which has the same contents; if you can, you have the answer as to which logical operation is being performed.

Repeat the process for `mystery1`, `mystery2`, and `mystery3`. Note, `mystery3` is a combination of several logical operators.

```
class Test {
  public static void main(String[] args) {
    BoolTest b = new BoolTest();
    b.go();
  }
}

class BoolTest {

  boolean mystery0(boolean x, boolean y) {
    if (x) return y;
    return false;
  }

  boolean mystery1(boolean x, boolean y) {
    if (x)
      if (y)
        return true;
    return false;
  }

  boolean mystery2(boolean x, boolean y) {
    if (x) return true;
    if (y) return true;
    return false ;
  }


  boolean mystery3(boolean x, boolean y) {
    if (x) return false;
    if (y) return true;
    return false ;
  }

  void go() {
    int num1 = 5 ;
    boolean result1 = mystery0(0 <= num1 , num1 <= 10);
    boolean result2 = mystery1(0 <= num1 , num1 <= 10);
    boolean result3 = mystery2(0 <= num1 , num1 <= 10);
    boolean result4 = mystery3(0 <= num1 , num1 <= 10);
    num1 = 11;
    boolean result5 = mystery1(0 <= num1 , num1 <= 10);
    boolean result6 = mystery2(0 <= num1 , num1 <= 10);
    boolean result7 = mystery3(0 <= num1 , num1 <= 10);
    System.out.println(result1 + " " + result2 + " " +
        result3 + " " + result4 + " " + result5 + " "
        + result6 + " " + result7);
  }
}
```

iv. What is the output of the program? **Hint:** for each method call, first evaluate the actual parameters and then make use of the truth tables developed in the previous part to determine the result.

2. Suppose we want to condition an if statement on whether two String objects, referenced by `stringOne` and `stringTwo`, are the same. What are the issues we should be aware of and what is the correct way to achieve this?

3. Assuming `numOne` and `numTwo` are integers, how many times might the following code print out "grape"? Explain your answer.

```
if(numOne > numTwo && numOne < numTwo)
   System.out.println("grape");
else {
   if(numOne == numTwo) {
     System.out.println("grape");
       if(numOne > numTwo)
         System.out.println("grape");
   }//end if numOne == numTwo
 }//end else
```

4. Write a short application that takes in a String from the user and prints it out backwards.

5. The following code compiles, but the program will appear to hang (e.g. nothing happens) if run. This is a clear sign of an infinite loop. What part of this code fragment would cause an infinite loop?

```
1 while(i < 50); {
2   System.out.println(i);
3   i += 2;
4 }
```

6. Write a short code fragment that uses a while loop to verify that the user enters a positive integer as input. You may assume that a Scanner object called `input` has already been instantiated.

7. What is wrong with the following code? Rewrite it so that it produces correct output.

```
if (total == MAX)
  if (total < sum)
    System.out.println ("total == MAX and < sum.");
else
  System.out.println ("total is not equal to MAX");
```

8. What is wrong with the following code fragment? Will this code compile if it is part of an otherwise valid program? Explain.

```
if (length = MIN_LENGTH)
System.out.println ("The length is minimal.");
```

9. What output is produced by the following code fragment?

```
int limit = 100, num1 = 15, num2 = 40;
if (limit <= limit) {
  if (num1 == num2)
    System.out.println ("lemon");
  System.out.println ("lime");
}
System.out.println ("grape");
```

10. What output is produced by the following code fragment?

```
int num = 1, max = 20;
while (num < max) {
  if (num % 2 == 0)
    System.out.println (num);
  num++;
}
```

11. What is wrong with the following code fragment? What are three distinct ways it could be changed to remove the flaw?

```
count = 50;
while (count >= 0) {
  System.out.println (count);
  count = count + 1;
}
```

12. Write a method called `evenlyDivisible` that accepts two integer parameters and returns `true` if the first parameter is evenly divisible by the second (or vice versa) and `false` otherwise. Return `false` if either parameter is zero.

13. Write a method called `isAlpha` that accepts a character parameter and returns `true` if that character is either an uppercase or lowercase alphabetic letter.

14. Write a method called `isIsoceles` that accepts three integer parameters that represent the lengths of the sides of a triangle. The method returns `true` if the triangle is isosceles but not equilateral (meaning that exactly two of the sides have an equal length), and `false` otherwise.