Workshop 2 (Week 3) - White Box Testing

The purpose of this workshop is to practice and develop an understanding of White Box testing.

1. Concepts

I. How many branches are there in an if statement? Why?

II. What about an else-if statement?

III. How many branches are there in a switch? Why?

IV. What does a loop look like in a Control Flow Diagram?

V. What does a Control Flow Diagram with multiple end points (i.e. return statements) look like?

2. Password Tester

Based on the documentation for the PasswordTester class (see Appendix B) and the source code file PassworTester.java (see Appendix A), answer the following questions:

I. Draw a Control Flow Diagram for the method PasswordTester.isStrong(String).

II. How many statements are there in the method?

III. How many branches are there?

IV. How many paths are there?

V. How many inputs are needed to cover all statements?

VI. How many inputs would needed to cover all branches?

VII. How many inputs would needed to cover all paths?

3. The Compute Median Example

Consider the following function that computes the Median value:

Task 1: Design some test cases for the Median function.

Task 2: Compute test coverage (including statement, branch, and path coverage).

Task 3: Design more test cases to achieve 100% statement, branch, and path coverage.

Task 4: Implement your test cases as jUnit test cases and execute the test cases.

(If jUnit is not installed at your PC, install it from: https://junit.org/, or follow the instructions here to install it from Maven https://www.jetbrains.com/help/idea/configuring-testing-libraries.html)

```
public static int median(int x, int y, int z){
        int median = 0;
        if(x >= y && x <= z){ // y<=x<=z
            median = x;
        } else if(x >= z && x <= y){ // z<=x<=y
            median = x;
        } else if(y >=x && y < z){ // x<=y<=z
            median = y;
        } else if(y >= z && y <= x){ // z<=y<=x
            median = y;
        } else {    // x<=z<=y or y<=z<=x
            median = z;
        }
        return median;
    }
}
```

4. Try the Web: Code In Game

https://www.codingame.com/ide/puzzle/the-descent

Appendix A: PasswordTester.Java

```
import java.util.regex.Pattern;
public class PasswordTester {
   public static boolean isStrong(String password) {
      boolean isStrong = true;
      if(password.length() < 8) {
         System.out.println("Notice: Your password has less than 8 characters.");
         isStrong = false;
      }
      if(!Pattern.compile("[a-z]").matcher(password).find()) {
         System.out.println("Notice: Your password does not contain a lower case letter.");
         isStrong = false;
      }
      if(!Pattern.compile("[A-Z]").matcher(password).find()) {
         System.out.println("Notice: Your password does not contain an upper case letter.");
         isStrong = false;
      }
      if(!Pattern.compile("[0-9]").matcher(password).find()) {
         System.out.println("Notice: Your password does not contain a number.");
         isStrong = false;
      }
      if(!Pattern.compile("[!@#\\$%\\^&\\*\\(\\)]").matcher(password).find()) {
         System.out.println("Notice: Your password does not contain a special.");
         isStrong = false;
      }
      if(isStrong) {
         System.out.println("Result: Strong password.");
      } else {
         System.out.println("Result: Weak Password.");
      }
      return isStrong;
   }
}
```

Appendix B: Class PasswordTester (in JavaDoc format)

public class **PasswordTester**
extends java.lang.Object

- ***Constructor Summary***

Constructors

Constructor and Description
PasswordTester()

- ***Method Summary***

All Methods **Static Methods** **Concrete Methods**

Modifier and Type	**Method and Description**
static boolean	isStrong(java.lang.String password) Tests the strength of a candidate password against several criteria.

- ***Constructor Detail***

- **PasswordTester**

public PasswordTester()

- ***Method Detail***

- **isStrong**

public static boolean isStrong(java.lang.String password)

Tests the strength of a candidate password against several criteria. For each criteria which has not been met, a notice will be displayed via standard output.
For the purposes of this test a strong password must have at least:
- A length of 8 or more characters
- 1 lower case letter
- 1 upper case letter
- 1 number
- 1 special character from the following list: !, @, #, $, %, ^, &, *, (,)

Parameters:

password - a password to test

Returns:

true if the password meets all of the above criteria, false otherwise.