Week 2 - Unit Testing

Testing is a critical part of software development that often involves executing a program to identify errors in the code.

There are many different software testing methodologies, techniques, and tools. We will focus on two basic approaches of software testing: black-box or white-box testing:

- **Black-box Testing:** Assesses program functionality based on requirements of what the program should do, without looking into the details of how the program is implemented. To perform black-box testing in CMSC 115, we will develop test cases that map a program input to an expected output, as well as test cases that map a method call to an expected return value.
- **White Box Testing:** Assesses program functionality based on its implementation. To perform white-box testing in CMSC 115, we will develop test cases to cover various execution paths through a program.

For the week#2 projects, we will perform basic black-box testing. Each test case will specify:

- the user input
- the expected output
- the actual output
- the test result

Example

Let's consider the following application:

REQUIREMENT: Write a program that prompts the user to enter the years (such as 1, 2, 1000, or 5000) and displays the equivalent number of minutes. For simplicity, assume a year has 365 days.

Notice that the requirements are a bit ambiguous in terms of the valid range of values, but it does list sample integer values for the number of years. The sample values will be used for the test cases as shown below.

> [!NOTE] The first two columns are filled out before writing the code, while the last two columns are filled out after writing and executing the code.

Input	Expected Output	Actual Output	Pass/Fail
1	525600 minutes		
2	1051200 minutes		
1000	525600000 minutes		
5000	2628000000 minutes		

Let's test with the following program:

```java
import java.util.Scanner;

public class YearsToMinutes {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter the number of years
        System.out.print("Enter the number of years: ");
        int years = input.nextInt();

        //Convert years to minutes
        int days =years * 365;
        int hours = days * 24;
        int minutes = hours * 60;

        System.out.println(minutes + " minutes");
    }

}
```

The tests are updated to record the actual output and result. Notice the last test fails to produce the correct result.

Input	Expected Output	Actual Output	Pass/Fail
1	525600 minutes	525600 minutes	Pass
2	1051200 minutes	1051200 minutes	Pass
1000	525600000 minutes	525600000 minutes	Pass
5000	2628000000 minutes	-1666967296 minutes	Fail