ASSIGNMENT 9

Generate one **invalid**, one **valid-but-not-useful**, and one **useful** mutants for the following program using the provided mutant operators. Assume that **a mutant is useful only if it is killed by at most a test case of the provided test suite**.

For your valid mutants **show the test cases in the provided test-suite that contribute to their killing**.

Generate one **equivalent** mutant, and explain why it is indeed equivalent.

Generate at least mutant that is **weakly killed, but not killed**, with the provided test-suite.

```
public class Date {
        private int year;
        private boolean leap;

        public Date(int y) {
                if (y <= 1584) {
                        throw new RuntimeException("Invalid year");
                }
                year = y;
                leap = (year % 400 == 0);
                leap = (leap || (year % 4 == 0 && year % 100 != 0));
        }

        public int lastDayOfMonth(int month) {
                switch (month) {
                case 1:
                case 3:
                case 5:
                case 7:
                case 8:
                case 10:
                case 12:
                        return 31;
                case 4:
                case 6:
                case 9:
                case 11:
                        return 30;
                case 2:
                        int max = 28;
                        if (leap) {
                                ++max;
                        }
                        return max;
                default:
                        throw new RuntimeException("Invalid month");
                }
        }
}
```

MUTANT OPERATORS:

1. A op B --> B op A, where op is a comparison operator
2. A op1 B --> A op2 B, where op1 and op2 are 2 different comparison operators
3. ++A --> A++
4. TRUE --> FALSE
5. FALSE --> TRUE
6. == --> =
7. = --> ==

TEST SUITE:

```
void test0() {
  Date d = new Date(2024);
  assertEquals(29, d.lastDayOfMonth(2));
}

void test1() {
  Date d = new Date(2023);
  assertEquals(31, d.lastDayOfMonth(3));
}

void test2() {
  Date d = new Date(2024);
  assertEquals(30, d.lastDayOfMonth(4));
}
```

INVALID MUTANT:

To generate an invalid mutant we can use the **mutant operator 6 (== --> =)** to change the leap of the code as follows:

```
public Date(int y) {
                if (y <= 1584) {
                        throw new RuntimeException("Invalid year");
                }
                year = y;
                leap = (year % 400 = 0);
                leap = (leap || (year % 4 == 0 && year % 100 != 0));
      }
```

This mutant is **invalid because it generates a compilation error** due to the assignment operator = being used instead of the equality operator ==.

VALID-BUT-NOT-USEFUL MUTANT:

To generate a valid-but-not-useful mutant we can use the **mutant operator 2 (A op1 B --> A op2 B)** to change the check of the year as follows:

```
public Date(int y) {

if (y >= 1584) {

throw new RuntimeException("Invalid year");

}

year = y;

leap = (year % 400 == 0);

leap = (leap || (year % 4 == 0 && year % 100 != 0));

}
```

This mutant is valid-but-not-useful because **test0(), test1() and test2() are all killed** because their year is greater than 1584.

USEFUL MUTANT:

To generate a useful mutant we can use the **mutant operator 2 (A op1 B --> A op2 B)** to change the leap of the code as follows:

```
    public Date(int y) {
                if (y <= 1584) {
                        throw new RuntimeException("Invalid year");
                }
                year = y;
                leap = (year % 400 == 0);
                leap = (leap || (year % 4 != 0 && year % 100 != 0));
          }
```

This mutant cause the leap variable to be set to true for non-leap years and false for leap years. This mutant will **not affect the test1() and test2()** since they do not involve the month of February so the mutant will not be killed in both cases. In this case, **the test0() would fail** because **it expects 29** as the last day of February 2024, however the mutant will cause the program to **return 28** as last day of the month which is false **causing the kill**.

EQUIVALENT MUTANT:

To generate an equivalent mutant we can use the **mutant operator 1 (A op B --> B op A)** to change the leap of the code as follows:

```
          public Date(int y) {
                      if (y <= 1584) {
                            throw new RuntimeException("Invalid year");
                      }
                      year = y;
                      leap = (year % 400 == 0);
                      leap = ((year % 4 == 0 && year % 100 != 0) || leap);
          }
```

The mutant is equivalent because the logical operator OR (||) is commutative, so **swapping the operands does not affect the execution of the program** and all the test suites will not generate the kill of the mutant itself.

WEAKLY KILLED BUT NOT KILLED MUTANT

To generate a weakly killed mutant that is not killed with the provided test-suite we can use the **mutant operator 2 (A op1 B --> A op2 B)** to change the check of the year as follows:

```
public Date(int y) {

if (y < 1584) {

throw new RuntimeException("Invalid year");

}

year = y;

leap = (year % 400 == 0);

leap = (leap || (year % 4 == 0 && year % 100 != 0));

}
```

This mutant is weakly killed because **the test suite does not contain a test case that has the year value equal to 1584**. The mutant **modified the functionality of the program** since an input with year 1548 would lead to an "Invalid year" error **but the test suite does not detect it thus making it a weakly killed mutant**.