Generate one invalid, one valid-but-not-useful, and one useful mutants for the following program using the provided mutant operators. For each mutant, show the test cases in the provided test-suite that contribute to its killing (This is pseudocode: assume that the == operator between string returns true if the two string are the same. Example: “ciao”==“ciao”-> true, “ciao” == “caiocaio” -> false).

For this exercise, assume that a mutant is useful only if it is killed by at most 1 given test case.

```
1 boolean f(String[] a, String b[]) {
2     for (int i = 0; i < a.length; i++) {
3         String s1 = a[i];
4         boolean found = false;
5         for (int j = 0; j < b.length; j++) {
6             String s2 = b[j];
7             if (s1 == s2) {
8                   found = true;
              }
          }
9         if (!found) {
10            return false;
          }
    }
11    return true;
}
```

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. c1 --> c2, where c1 and c2 are two different constants
5. TRUE --> FALSE
6. FALSE --> TRUE
7. == --> =
8. = --> ==

TEST SUITE:

1. [“software”], [“software”] --> TRUE
2. [“software”, “quality”], [“software”] --> FALSE
3. [“usi”, “quality”, “software”], [“software”, “quality”] --> FALSE

INVALID MUTANT:

In line 7 change the operator s1==s2 to s1>s2. This is invalid because you cant compare strings with a > or < operator.

The test case will result in an error since the comparison operator is not suited for string comparison.

VALID BUT NOT USEFUL MUTANT:

in line 5 change the code with for (int j = b.length - 1; j >= 0; j--). This change only change the order which the strings are considered so it is not usefull.

The execution of test cases will procced without any killing because the logic of the program is not changed in a significant way.

USEFUL MUTANT:

In line 5 change the code with for (int j = 0; j != b.length - 1; j++). In this way the last element of the b array wont be considered.

The execution of test case 2 will kill the mutant:

Input:

a = ["software", "quality"]

b = ["software"]

Expected output:

false

Actual output from original program:

false

Actual output from mutant program:

true