# Piacente Cristian 866020

## Homework H12 – Regression Testing

### Software Quality, Academic Year 2023-2024, University of Milan – Bicocca

Consider the following specification of the program below.

Spec: Program deleteIfEqual removes items from a set of strings represented as an array. It shall return the number of items that were indeed removed. The program shall check that the provided array is not null, not empty and includes a maximum of 100 items; if the array is null, the program shall throw an exception; for arrays larger than the maximum size, it shall return the error code -100; for empty arrays it shall return the error code -1.

```
int deleteIfEqual(String[] theSet, String toBeDeleted) {
    if (theSet == null) {
        throw new RuntimeException("Invalid set");
    }
    if (theSet.length > 100) {
        return -1000;
    }
    int count = 0;
    for (int i = 0; i < theSet.length; i++) {
        if (theSet[i] != null && theSet[i].equals(toBeDeleted)) {
            theSet[i] = null;
            count++;
        }
    }
    return count;
}
```

Your tasks are:

1) Refer to any test section approach of your choice (functional, structural, or even better a combination of both) to identify a test suite for the program. Briefly explain the criterion/criteria and document the selected test cases. Then complete each test case with concrete inputs and proper oracles based on the provided spec.

2) Compile a test report that indicates which test cases pass and fail, respectively, for the program indicated in the exercise. (You do not need to physically execute the test cases, it is enough to evaluate the results based on the provided code.)

3) If the execution of the test cases revealed some failures (likely it should if your test cases are adequate), propose a fixed version of the program for which all your test cases pass.

4) With reference to the new version of the program that you obtained as result of task 3, say which test you should re-run for regression testing if you work with the following code-based test selection approach: "re-run the test cases that execute new or modified assignments, or new or modified return statements."

**TASK 1**

Let's choose a combined approach to exploit both functional and structural testing: functional w.r.t the specification and structural w.r.t the given implementation.
We will combine **boundary condition testing** (functional) with the **branch coverage criterion** (structural).

To apply boundary condition testing, let's start with defining the inputs, parameters and results.
Inputs: **array of strings** (theSet), **string to be deleted** (toBeDeleted)
Parameters: none (no environment variables)
Results: **count of deleted items**

Let's perform boundary condition testing based on the specification:

- **Input array of strings: reference**
  - Null [Error]
  - Not null
- **Input array of strings: length**
  - 0 [Property EMPTY]
  - 1

  - > 1 and < 100
  - 100
  - > 100 [Property TOO_LARGE]
- **Input string to be deleted: number of occurrences in array of strings**
  - 0
  - 1
  - > 1
- **Result count of deleted items**
  - -100 [Single] [IF TOO_LARGE]
  - -1 [Single] [IF EMPTY]
  - Number of occurrences

Please note that we didn't check if the string to be deleted is null because we can assume there is a sanity check that makes sure it is not null, so by pre-condition the string to be deleted is always not null.

We can now create test cases using concrete inputs, considering the created combinations:

| ID | theSet | toBeDeleted | Oracle |
|---|---|---|---|
| 1 | null | "a" | Exception |
| 2 | {"a" 101 times} | "a" | -100 |
| 3 | {} | "a" | -1 |
| 4 | {"a"} | "a" | 1 |
| 5 | {"a"} | "b" | 0 |
| 6 | {"a", "a"} | "a" | 2 |
| 7 | {"a", "b"} | "a" | 1 |
| 8 | {"a", "a"} | "b" | 0 |
| 9 | {"a" 100 times} | "a" | 100 |
| 10 | {"b" followed by "a" 99 times} | "b" | 1 |
| 11 | {"a" 100 times} | "b" | 0 |

Let's now analyze the implementation, using branch coverage criterion, to eventually add more test cases.

- if (theSet == null)
  Both branches are already covered by the test cases above (e.g. see test #1 and #2).
- if (theSet.length > 100)
  Both branches are already covered by the test cases above (e.g. see test #2 and #3).
- For loop
  It is executed zero times by test case #3 and at least once by some of the other tests (e.g. see test #4).
- if (theSet[i] != null && theSet[i].equals(toBeDeleted))
  Both branches are already covered by the test cases above (e.g. see test #7).

It follows that our test suite **already satisfies branch coverage criterion**.

**TASK 2**

Let's generate a report for the test cases, to indicate which pass and which fail ("Result" refers to the result of the execution, while "Evaluation" is PASS if the oracle matches the result, FAIL otherwise):

| ID | theSet | toBeDeleted | Oracle | Result | Evaluation |
|---|---|---|---|---|---|
| 1 | null | "a" | Exception | Exception | PASS |
| 2 | {"a" 101 times} | "a" | -100 | -1000 | FAIL |
| 3 | {} | "a" | -1 | 0 | FAIL |
| 4 | {"a"} | "a" | 1 | 1 | PASS |
| 5 | {"a"} | "b" | 0 | 0 | PASS |
| 6 | {"a", "a"} | "a" | 2 | 2 | PASS |
| 7 | {"a", "b"} | "a" | 1 | 1 | PASS |
| 8 | {"a", "a"} | "b" | 0 | 0 | PASS |
| 9 | {"a" 100 times} | "a" | 100 | 100 | PASS |
| 10 | {"b" followed by "a" 99 times} | "b" | 1 | 1 | PASS |
| 11 | {"a" 100 times} | "b" | 0 | 0 | PASS |

# Piacente Cristian 866020

## Homework H12 – Regression Testing

Software Quality, Academic Year 2023-2024, University of Milan – Bicocca

TASK 3

To make all the test cases pass, we must change the implementation as follows:

- if theSet.length > 100, then -100 must be returned **instead of -1000**
- add the following if statement:
  ```
  if (theSet.length == 0) {
      return -1;
  }
  ```

Here is the **fixed version of the program**, for which all the test cases pass:

```
int deleteIfEqual(String[] theSet, String toBeDeleted) {
    if (theSet == null) {
        throw new RuntimeException("Invalid set");
    }
    if (theSet.length > 100) {
        return -100;
    }
    if (theSet.length == 0) {
        return -1;
    }
    int count = 0;
    for (int i = 0; i < theSet.length; i++) {
        if (theSet[i] != null && theSet[i].equals(toBeDeleted)) {
            theSet[i] = null;
            count++;
        }
    }
    return count;
}
```

---

TASK 4

For regression testing, working with the approach "re-run the test cases that execute new or modified assignments, or **new or modified return statements**", we should re-run the tests **#2** and **#3**, since they are the only ones which are affected by the new modifications (test #2 executes the modified return statement and test #3 executes the new return statement).