**Test Plan for Ataccama TestCalc application**

**Test Description**

* Performing math operations (addition, subtraction, division, multiplication).
* Verify absence of errors and exceptions in case of valid input values.
* Verify stability of the application.

Record operation result. Compare with the result from the external calculator (e.g. Windows Calculator).

**Test Environment**

* Install Apache Tomcat web server locally.
* Download the testCalc.war file and place it in the Tomcat\webapps\ folder.
* Run the Apache Tomcat web server.
* Open browser, go to URL <http://localhost:8080/testCalc/>.

**Test parameters**

**Value1:** integer values from -2147483648 to 2147483647

**Value2:** integer values from -2147483648 to 2147483647

**Operations:** addition, subtraction, multiplication, division

The main challenge is to choose a valid set of test values to cover all risky combinations. To best way to achieve this is to use such Test Design techniques as Equivalence Partitioning and Boundary Value Analysis.

All data can be divided into the following equivalence classes:

1. **Below zero**. Values from -2147483648 to 1.
2. **Zero**. Contains only one value – 0.
3. **Above zero**. Values from 1 to 2147483647.
4. Integers with **leading zero**, e.g. 0000012345689 – zeroes should be removed automatically.

In addition, there can be considered class with negative, which contain values, causing the exception in the calculator.

1. Negative.

* -2147483649, 2147483648 – values out of integer range
* empty values
* non-integer values (e.g. 1.0)
* strings (e.g. “aBd%##@(%)@”)
* valid integers with trailing empty strings, e.g. “1111111 “
* zero division

Final classes:

1. **Below zero**:

-2.147.483.648 boundary value

-46.340

-215 with leading zero

-15

-1 boundary value

1. **Above zero**:

2.147.483.647 boundary value

46.340

215

15 with leading zero

1. boundary value
2. **Zero:** 0
3. **Negative**:

-2147483648 Integer.MIN\_VALUE - 1

2147483648 Integer.MAX\_VALUE + 1

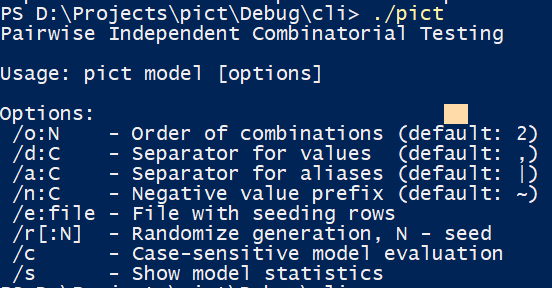
99.0 Non-integer

!@#$%^&\*() as8o5 String

123 34 3 Invalid number with spaces

“ “ Empty string

For decreasing the number of test combinations was used **Pairwise Testing** technique.

Pairwise testing (a.k.a. all-pairs) is an effective test case generation technique that is based on the observation that most faults are caused by interactions of at most **two** factors.

Pairwise-generated test suites cover **all combinations** of two therefore are much smaller than exhaustive ones yet still very effective in finding defects.

For generating test parameters combinations was used tool PICT (Pairwise Independent Combinatorial Testing tool) by Microsoft

**PICT model for positive combinations**:

**Operations**: addition, subtraction, multiplication, division

**Value1**: -2147483648, -46340, 215, 15, 1, 0, 2147483647, 46340, 215, 1

**Value2**: -2147483648, -46340, 215, 15, 1, 0, 2147483647, 46340, 215, 1

Combinations: 180

Generated tests:100 (test number was reduced after removing some invalid values, e.g. zero-division).

**Test data sample**

|  |  |  |
| --- | --- | --- |
| **Operations** | **Value1** | **Value2** |
| addition | 215 | 15 |
| addition | 0 | 1 |
| addition | 2147483647 | 215 |
| addition | 1 | 1 |
| addition | 46341 | 2147483647 |
| addition | 15 | 46341 |
| addition | 215 | 215 |
| addition | -46340 | 0 |
| addition | 215 | 0 |
| addition | 2147483647 | -46340 |
| addition | 1 | 15 |
| addition | 215 | -2147483648 |
| addition | -2147483648 | -46340 |
| addition | 215 | 215 |
| addition | 215 | 1 |
| addition | 1 | -46340 |
| addition | -46340 | 1 |
| addition | 0 | 0 |
| addition | 1 | 0 |
| addition | 215 | 2147483647 |
| addition | -46340 | 15 |
| addition | 1 | 1 |
| addition | 1 | 15 |
| addition | 0 | 2147483647 |
| addition | -2147483648 | 215 |
| addition | -2147483648 | 1 |
| addition | 215 | 15 |
| addition | -46340 | 215 |
| division | 0 | 215 |
| division | 2147483647 | 1 |
| division | 15 | 215 |
| division | 46341 | 15 |
| division | 215 | 1 |
| division | 0 | -46340 |
| division | -2147483648 | 15 |
| division | -46340 | -2147483648 |
| division | 2147483647 | 1 |
| division | 46341 | -46340 |
| division | 215 | -2147483648 |
| division | 215 | 215 |
| division | -46340 | 2147483647 |
| division | 46341 | 1 |
| division | 46341 | -2147483648 |
| division | -46340 | 1 |
| division | 15 | 215 |
| multiplication | 1 | 0 |
| multiplication | 46341 | 215 |
| multiplication | 0 | -2147483648 |
| multiplication | 1 | -46340 |
| multiplication | 215 | 1 |

**Test cases examples**

**\* \* \***

**TC 1** (see Table 1, #1)

*Preconditions*:

* OS: Windows 10.
* Installed applications: MS Office, Skype, Telegram, Chrome, Firefox.
* Mounted USB drive for archive logs saving.
* Adjust system date to current + 6 days, reboot the VM (for logs age limit testing).
* Generated programs activity after reboot.

*Steps*:

1. Launch the logcollector.exe
2. **Select all** artifacts to collect
3. Logs age limit – select **5** days
4. Logs collections mode – select “**Original binary from disk**”
5. Save archive as – choose **USB** drive
6. File name – leave **predefined**
7. Press “**Collect button**”
8. Go to USB drive, open log file
9. Compare the information in log with actual system information (Task Manager, System Configuration, etc)
10. Check the oldest entries in log

*Expected Results*:

1. Application launched successfully
2. All checkboxes **selected**
3. Logs age limit **selected**
4. Logs collections mode **selected**
5. **Path** to archives populated successfully
6. **Default** file name specified
7. Program works without errors and exceptions. **Operations log** has new entries according to program activity
8. Log file saved on the **USB** drive. File opens successfully
9. Information in log is correct and equals with the information about the system from Task Manager, System Configuration, Event viewer.
10. There are no entries in log file older than **5** days ago.