Functional Testing Justification

Farha Jawed

Testing Sqrt:

Boundary value analysis has been applied while testing the square root function. Inputs less than 0 come under invalid class. Inputs in the range of 0 to 2147483647 (considering 32bit integer storage) fall under valid class. Inputs at exact boundary (0), one below boundary (-1) and one above boundary (1) are tested at lower range. Along with that several small, medium and large positive numbers including the largest possible 32bit integer number (2147483647) at upper boundary and a negative invalid number well outside the boundary are tested making the implementation justifiable.

Testing Sqr:

Boundary value analysis has been applied for testing the implementation. Boundaries between equivalence partitions have been tested. There are 2 invalid and 1 valid input classes.

Invalid classes: Numbers greater than 46340 (i.e. square root of largest possible 32bit int) and smaller than -46340.

Valid classes: Numbers in the range of -46340 to 46340.

Each number at exact boundary, one below boundary and one above boundary are tested for both upper and lower ranges. Along with that, invalid numbers well above boundaries are tested. Small, medium and large valid numbers are present in test suite. All test cases showed desired behavior making the implementation acceptable.

Testing Factorial:

Boundary value analysis has been applied for testing the implementation. Boundaries between equivalence partitions have been tested. There are 2 invalid and 1 valid input classes.

Invalid classes: Numbers greater than 12 (since factorial of number greater than 12 surpasses largest possible 32bit integer) and smaller than 0.

Valid classes: Numbers in the range of 0 to 12.

Each number at exact boundary, one below boundary and one above boundary are tested for both upper and lower ranges. Along with that, invalid numbers well above and below upper and lower boundaries are tested respectively. Several valid inputs present at center are tested. All test cases showed desired behavior making the implementation acceptable.

Testing Sumup:

Like above cases, boundary value analysis has been applied for testing the implementation of sum up. Boundaries between equivalence partitions have been tested. There are 2 invalid and 1 valid input classes.

Invalid classes: Numbers less than 0 and greater than 65535 (since summation of 0 to number above 65535 overflows 32bit integer storage)

Valid classes: Numbers in the range of 0 to 65535.

Each number at exact boundary, one below boundary and one above boundary are tested for both upper and lower ranges. Along with that, invalid numbers well above and below upper and lower boundaries are tested respectively. Several valid inputs present at center are tested. All test cases showed desired behavior making the implementation acceptable.

Testing simpleFunctionXplusY:

Considering X as the first number and Y as the second number, I checked for possible combinations:

(X+Y)

(X)+(-Y)

(-X)+Y

(-X)+(-Y)

X+0

0+Y

0+0 (X and Y both 0)

For each of these combinations, output above 2147483647 and below -2147483648 will create overflow. Several valid data for small, medium, large input was tested. Also, boundary value analysis was used to check boundary values for both valid and invalid inputs. All test cases showed desired behavior making the implementation acceptable.

Testing Despacer:

Several possibilities of valid cases have been considered:

1. Input without any space and text

2. Input with one space without text

3. Input with multiple space without text

4. Input text without any space

5. Input text with one space between strings

6. Input text with multiple spaces between strings

7. Input text with multiple spaces at the beginning and/or at the end

Test cases passed for all the above possible scenario thus making the implementation acceptable.