1. Test cases

a) Takes an integer as input and determines if it lies in the range: -10 <= x <=10.

* Input of -10 and Input of 10 (boundary values, Return true)
* Input of -11 and Input of 11 (one value beyond boundary values, Return false)
* Input of -9 and Input of 9 (one value under boundary values, Return true)
* Input of 0 (Return true)
* MAX\_INT (Return false)
* MIN\_INT(Return false)

|  |  |  |
| --- | --- | --- |
| Unit Name | boundary-check | |
| Summary (e.g. Interface spec) | Takes an integer as input and determines if it lies in the range: -10 <= x <=10. | |
| Input Type(s) | Input Value(s) | Expected Result |
| Negative integer | -10 | true |
| Positive integer | 10 | true |
| One value less than lower boundary | -11 | false |
| One value greater than upper boundary | 11 | false |
| One value greater than lower boundary | -9 | true |
| One value less than upper boundary | 9 | true |
| Zero value | 0 | true |
| MAX\_INT | Integer.MAX\_VALUE | false |
| MIN\_INT | Integer.MIN\_VALUE | false |

b) Takes a float as input and determines if it lies in the range: (x < -10) or (x > 10).

* Input of -11.0 and input of 11.0 (lower boundary values, Return true)
* Input of -10.0 and input of 10.0 (one value less that lower boundary values, Return false)
* Input of -12.0 and input of 12.0 (one value greater than lower boundary values, Return true)
* Input of 0 (Return false)
* Input of -20 and input of 20 (Return true)
* Positive and Negative infinity (Float.***POSITIVE\_INFINITY*** and Float.***NEGATIVE\_INFINITY*** , Return true)
* MAX\_Float (Float.MAX\_VALUE, Return true)
* NaN (Float.NaN, Return false)
* MIN\_Float (Float.MIN\_VALUE, Return false)

|  |  |  |
| --- | --- | --- |
| Unit Name | boundary-check | |
| Summary (e.g. Interface spec) | Takes a float as input and determines if it lies in the range: (x < -10) or (x > 10). | |
| Input Type(s) | Input Value(s) | Expected Result |
| Negative float(lower boundary) | -11 | true |
| Positive float(lower boundary) | 11 | true |
| One value greater than negative lower boundary | -10 | false |
| One value less than positive lower boundary | 10 | false |
| One value less than negative lower boundary | -12 | true |
| One value greater than upper boundary | 12 | true |
| Zero value | 0 | false |
| Negative float | -20 | true |
| Positive float | 20 | true |
| Positive infinity | Float.POSITIVE\_INFINITY | true |
| Negative infinity | Float.NEGATIVE\_INFINITY | true |
| MAX FLOAT | Float.MAX\_VALUE | true |
| NaN | Float.NaN | false |
| MIN FLOAT | Float.MIN\_VALUE | false |

c) Takes in a 10 digit telephone number as a string and returns true if the string is a legitimate phone number and false otherwise.

Legitimate means:

. properly formed (10 digits)

. does not begin with a 1 or 0

. the first three digits cannot be 555

c) Takes in a 10 digit telephone number as a string and returns true if the string is a legitimatephone number and false otherwise.

Legitimate means:

. properly formed (10 digits)

. does not begin with a 1 or 0

. the first three digits cannot be 555

|  |  |  |
| --- | --- | --- |
| Unit Name | telephoneNumber | |
| Summary (e.g. Interface spec) | Checks if the telephone number is legitimate or not | |
| Input Type(s) | Input Value(s) | Expected Result |
| Properly formed | “8574138932” | true |
| Does not begin with 555 | “9941232528” | true |
| Less than 10 digits | “895642” | false |
| Begins with 1 contains 10 digits | “185743893” | false |
| Begins with 0 and contains 10 digits | “085741393” | false |
| Begins with 555 and contains 10 digits | “5558574138” | false |
| Begins with 555 and does not contain 10 digits | “5558574” | false |
| Begins with 1 and does not contain 10 digits | “155278” | false |
| Begins with 0 and not properly formed | “0857413” | false |
| More than 10 digits | “85741389324” | false |
| Contains 9 digits | “857413893” | false |
| Contains symbols | “+185741389” | false |
| Null | null | false |
| Empty | ‘ ‘ | false |
| String | “Mikeweintr” | false |
| Non character | “@” | false |

2. Test Specification Template for sqrt (homework):

|  |  |  |
| --- | --- | --- |
| Unit Name | sqrt (Interface- homework) | |
| Summary (e.g. Interface spec) | produces the square root of a non-negative input number n | |
| Input Type(s) | Input Value(s) | Expected Result |
| Non negative number | 1 | 1 |
| Zero input | 0 | 0 |
| Negative number | -1 | IllegalArgumentException |
| Non negative number | 3 | 1.7320508075688772 |
| Non negative number | 100 | 10.0 |
| MAX\_INT | Integer.MAX\_VALUE | 46340.950001051984 |
| MAX\_INT + 1 (one value above MAX\_INT) | Integer.MAX\_VALUE+1 | Overflow exception / IllegalArgumentException |
| MAX\_INT-1 | Integer.MAX\_VALUE-1 | 46340.949990262394 |
| MIN\_INT | Integer.MIN\_VALUE | IllegalArgumentException |

Test Specification Template for sqr (homework):

|  |  |  |
| --- | --- | --- |
| Unit Name | sqr (Interface- homework) | |
| Summary (e.g. Interface spec) | produces the square of an input number n | |
| Input Type(s) | Input Value(s) | Expected Result |
| Positive integer | 1 | 1 |
|  | 0 | 0 |
| Negative integer | -1 | 1 |
| Non negative number | 3 | 9 |
| Negative integer | -3 | 9 |
| MIN\_INT | Integer.MIN\_VALUE | Overflow Exception |
| MAX\_INT | Integer.MAX\_VALUE | Overflow Exception |
| Max Integer value which can be squared | 46340 | 2147395600 |
| One value above Max integer value which can be squared | 46341 | Overflow exception |
| One value below Max integer value which can be squared | 46339 | 2147302921 |
| Min Integer value which can be squared | -46340 | 2147395600 |
| One value below Min integer value which can be squared | -46341 | Overflow exception |
| One value above Min integer value which can be squared | -46339 | 2147302921 |

Test Specification Template for factorial (homework):

|  |  |  |
| --- | --- | --- |
| Unit Name | factorial (Interface- homework) | |
| Summary (e.g. Interface spec) | produces n! | |
| Input Type(s) | Input Value(s) | Expected Result |
| Non negative integer | 1 | 1 |
| Zero | 0 | 1 |
| Negative integer | -1 | IllegalArgumentException |
| Non negative number | 3 | 6 |
| Negative integer | -3 | IllegalArgumentException |
| Non negative number | 10 | 3628800 |
| MIN\_INT | Integer.MIN\_VALUE | IllegalArgumentException |
| MAX\_INT | Integer.MAX\_VALUE | Overflow Exception |
| MAX Value for Integer factorial | 12 | 479001600 |
| One value above Max value for Integer Factorial | 13 | Overflow exception |
| One value below Max value for Integer factorial | 11 | 39916800 |

3. I will reject the test specification because (1,-MAX\_INT) is a valid computation which would return **-2147483646** and it is not an Underflow Exception. The test should include input value of (-1,MIN\_INT) for Underflow exception. The tests can also include a positive and negative combination of input and inputs which would result in 0.

|  |  |  |
| --- | --- | --- |
| Unit Name | Adder | |
| Summary (e.g. Interface spec) | Takes two integers as input and returns their sum(as an integer) | |
| Input Type(s) | Input Value(s) | Expected Result |
| Positive integer, Positive integer | 1,1 | 2 |
| Positive integer, Positive integer | 5,35 | 40 |
| Positive large number, Positive large number | 100936,1327927 | 1428863 |
| Negative integer, negative integer | -3,-5 | -8 |
| Negative large integer, Negative large integer | -100936,-1327927 | -1428863 |
| 1, MAX\_INT | 1, Integer.MAX\_VALUE | Overflow Exception |
| 1, -MAX\_INT | 1, -Integer.MAX\_VALUE | -2147483646 |
| -1 , MIN\_INT | -1, Integer.MIN\_VALUE | Underflow Exception |
| 1, MIN\_INT | 1, Integer.MIN\_VALUE | -2147483647 |
| -1, MAX\_INT | -1, Integer.MAX\_VALUE | 2147483646 |
| MAX\_INT, MIN\_INT | Integer.MAX\_VALUE, Integer.MIN\_ | -1 |
| Positive, Negative integer | -1,1 | 0 |
| Positive, Negative integer | 3,-5 | -2 |
| Positive , Negative integer | 5,-3 | 2 |