HW2: CS5500

Shashwat Sanghavi

Test Specification for functional units

Unit Name		Sqrt		
Summary (e.g. Interface spec)		@param n the radicand @return sqrt of the radiant		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Notes
1	Int	Int.MIN_VALUE	NaN	Sqrt is not defined for negative numbers
2	Int	Int.MAX_VALUE	46340.950001051984	Positive case
3	Int	Int.MIN_VALUE-1	NaN	Sqrt is not defined for negative numbers
4	Int	Int.MAX_VALUE-1	46340.949990262394	Positive case
5	Int	Int.MAX_VALUE+1	NaN	Boundary, Overflow
6	Int	0	0	Positive case
7	Int	1	1	Positive case
8	Int	4	2	Positive case
9	Int	23423423	4839.77509808049	Positive case
10	Int	-23423423	NaN	Sqrt is not defined for negative numbers
11	Int	-9	NaN	Sqrt is not defined for negative numbers
12	Int	Null	Runtime Exception	Error
13	Float/ Double	4.5	Runtime Exception	Error

14	Str	3123bh	Runtime Exception	Error

Test case 1,2,3,4,5 checks the boundary case. 1,3 expects NaN as sqrt is not defined for negative numbers. 2,4 checks possible answer for the input. 5 should return NaN because of the overflow as adding one to the maximum possible integer would overflow.

6,7,8,9 are positive test cases which expects square root for the input numbers. These are the actual cases for which the function is expected to work properly. 9 checks the result for large positive number. 10,11 checks output for the negative input and should return NaN as sqrt is not defined for negative numbers which is a negative test-case. 12 checks for NULL input which should not be able to parse by integer and thus would result into runtime exception. 13, 14 checks for incorrect input type and expects runtime exception because the function expects only and only integer as an input type. So any other type of input for this function should result into runtime exception,

Thus the given test suite checks boundary cases, negative inputs, positive inputs, error terms, small numbers, large numbers and different input types.

Unit Name		Sqr		
Summary (e.g. Interface spec)		@param n the factor @return sqrt of the radiant		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Notes
1	Int	Int.MIN_VALUE	Int.MAX_VALUE	Overflow
2	Int	Int.MAX_VALUE	Int.MAX_VALUE	Overflow
3	Int	0	0	Positive case
4	Int	1	1	Positive case
5	Int	4	16	Positive case
6	Int	2342	5484964	Positive case
7	Int	-2342	5484964	Positive case
8	Int	-9	81	Positive case
9	Int	Null	Runtime exception	Error
10	Float/ Double	4.5	Runtime exception	Error

11	Str	3123bh	Runtime excception	Error
12	Int	46342	Int.MAX_VALUE	Overflow, Error case
13	Int	46340	2147395600	Positive case

Test case 1, 2 checks boundary conditions. This should return max_value as the output for max integers and min integer is larger than the max integer.

Case 3,4,5,6,7 and 9 are positive test cases. This cases checks the correct behavior of the function. 6 checks positive input which returns a large output and 7 checks a negative input which returns large output. In 8, we check a case where input is small. 9 checks for NULL input which should not be able to parse by integer and thus would result into runtime exception. 10, 11 checks for incorrect input type and expects runtime exception because the function expects only and only integer as an input type. So any other type of input for this function should result into runtime exception.

12 checks for a large input integer which results into overflow. 13 checks a large number that leads to an answer closest to the boundary.

Thus the given test suite checks boundary cases, negative inputs, positive inputs, error terms, small numbers, large numbers and different input types.

Unit Name		Factorial		
Summary (e.g. Interface spec)		@param n the largest factor to consider @return n!		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Notes
1	Int	Int.MIN_VALUE	NaN	Factorial is defined only for nonnegative numbers
2	Int	Int.MAX_VALUE	StackOverFlow Exception	OverFlow
3	Int	Int.MIN_VALUE-1	NaN	Factorial is defined only for nonnegative numbers
4	Int	Int.MAX_VALUE-1	StackOverFlow Exception	OverFlow
5	Int	Int.MAX_VALUE+1	StackOverFlow Exception	OverFlow
6	Int	0	0	Positive Case
7	Int	1	1	Positive Case

8	Int	4	24	Positive Case
9	Int	16	2004189184	Positive Case
10	Int	17	-288522240	Overflow
11	Int	-9	NaN	Factorial is defined only for nonnegative numbers
12	Int	Null	Runtime Error	Error
13	Float/ Double	4.5	Runtime Error	Error
14	Str	3123bh	Runtime Error	Error

Test case 1,2,3,4,5 checks for the boundary conditions. As factorial is not defined for the negative numbers, it excepts NaN in cases 1,3. Due to StackSize restrictions it raises stackOverflow exception in 2,4,5.

6,7,8,9 are the positive cases for which the factorial function works properly. Case 9 checks for an input for which it generates output closest to the boundary. In case 10 it overflows as the result is greater than the maximum value for an integer. It return NaN in case 11 as factorial is defined only for nonnegative integers. 12 checks for NULL input which should not be able to parse by integer and thus would result into runtime exception. 13, 14 checks for incorrect input type and expects runtime exception because the function expects only and only integer as an input type. So any other type of input for this function should result into runtime exception.

Thus the given test suite checks boundary cases, negative inputs, positive inputs, error terms, small numbers, large numbers and different input types.

Unit Name		SumUp		
Summary (e.g. Interface spec)		@param n the largest addend @return the sum		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Notes
1	Int	Int.MIN_VALUE	NaN	SumIp is defined only for nonnegative numbers
2	Int	Int.MAX_VALUE	-1073741824	OverFlow

3	Int	Int.MIN_VALUE-1	NaN	SumIp is defined only for nonnegative numbers
4	Int	Int.MAX_VALUE-1	-1073741823	OverFlow
5	Int	Int.MAX_VALUE+1	-1073741823	OverFlow
6	Int	0	0	Positive Case
7	Int	1	1	Positive Case
8	Int	4	10	Positive Case
9	Int	16	136	Positive Case
10	Int	46341	-1073716337	Overflow
11	Int	-9	NaN	SumIp is defined only for nonnegative numbers
12	Int	Null	Runtime Error	Error
13	Float/ Double	4.5	Runtime Error	Error
14	Str	3123bh	Runtime Error	Error

Test case 1,2,3,4,5 checks for the boundary conditions. As SumUp is not defined for the negative numbers, it excepts NaN in cases 1,3. Due to maximum value restrictions it overflows in 2,4,5.

6,7,8,9 are the positive cases for which the SumUp function works properly. In case 10 it overflows as the result is greater than the maximum value for an integer. It return NaN in case 11 as the function is defined only for nonnegative integers. 12 checks for NULL input which should not be able to parse by integer and thus would result into runtime exception. 13, 14 checks for incorrect input type and expects runtime exception because the function expects only and only integer as an input type. So any other type of input for this function should result into runtime exception.

Thus the given test suite checks boundary cases, negative inputs, positive inputs, error terms, small numbers, large numbers and different input types.

Unit Name	simpleFunctionXplusY

Summary (e.g. Interface spec)		@param x the first addend @param y the second addend @return the sum of x and y		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Notes
1	Int, Int	X: Int.MIN_VALUE, Y:Int.MIN_VALUE	0	Overflow
2	Int, Int	X: Int.MAX_VALUE, Y:Int.MAX_VALUE	-2	Overflow
3	Int, Int	X: Int.MIN_VALUE, Y:Int.MAX_VALUE	-1	Positive Case
4	Int, Int	X: Int.MIN_VALUE, Y: -1	2147483647	Overflow
5	Int, Int	X: Int.MIN_VALUE, Y: 1	-2147483647	Positive case
6	Int, Int	X: Int.MAX_VALUE, Y: -1	2147483646	Positive case
7	Int, Int	X: Int.MAX_VALUE, Y: 1	-2147483648	Overflow
8	Int, Int	X: 5 Y: 4	9	Positive Case
9	Int, Int	X: -5 Y: 4	-1	Positive Case
10	Int, Int	X: -5 y: -4	-9	Positive Case

11	Int, Int	X: 0 y: 0	0	Positive Case
12	Int, Int	X: Null Y: Null	Runtime Exception	Error
13	Int, Int	X: 5 y: Null	Runtime Exception	Error
14	Str, Int	X: 3123bh Y: 33	Runtime Exception	Error
15	Float, Int	X: 33.3 Y: 22	Runtime Exception	Error
16	Str, Float	X: 33.3df Y: 34.33	Runtime Exception	Error
17	int,int	X: Int.MAX_VALUE/2, Y: Int.MAX_VALUE/2	Int.MAX_VALUE	Boundary
18	int, int	X: Int.MAX_VALUE+1 Y:Int.MAX)VALUE+1	Int.MAX_VALUE	Overflow

Case 1,2,3,4,5,6,7 checks boundary conditions. Case 1 2 and 3 checks conditions where both the numbers are either same or different boundaries. When both the numbers are same boundaries, addition of them would result into overflow. However when numbers are different boundaries it should result into -1. In case 4, when a negative number is added to min value, it should overflow. In case 5, when positive number is added to a negative number, it results into an integer between min value and max value. When a positive number is added to the max value, it overflows and when a negative number is added to the max value, it results into a number between min and max value.

In case 8 two positive numbers are added. In case 9 a positive and a negative numbers are aded and in case 10 two two negative numbers are added.These should return an answer between min and max value. Case 16 checks a positive case that leads to a boundary. In case 18, the result is greater than the max value and thus it is a boundary condition.

For these cases, the value of X and Y can be replaced.

12,13 checks for NULL input which should not be able to parse by integer and thus would result into runtime exception. 14, 15, 16 checks for incorrect input type and expects runtime exception because the function expects only and only integer as an input type. So any other type of input for this function should result into runtime exception.

Thus the given test suite checks boundary cases, negative inputs, positive inputs, error terms, small numbers, large numbers and different input types.

Unit Name		Despacer		
Summary (e.g. Interface spec)		@param inputText the input text @return the string with only single spaces in it		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Notes
1	String	“”	“”	Positive Case
2	String	“Hi hello ”	“Hi hello “	Positive Case
3	String	“Hi hello”	“Hi hello”	Positive Case
4	String	Null	Runtime Exception	Error
5	String	“hello “	“hello “	Positive Case
6	String	“hello\t\thi\t”	“hello\t\thi\t”	Positive Case
7	String	“hello\t \thi\t”	“hello\t hi\t”	Positive Case
8	Int	4	Runtime Exception	Error
9	Float	16.0	Runtime Exception	Error
10	String	“ “	“ “	Positive case

Test case 1 checks an empty string. It expects same empty string as an output as there are no spaces in it. In case 2 there are multiple continuous spaces, it expects a string with single spaces as output.

Case 4 checks the null input. The functions expects non null input and thus throws runtime exception. 5,6,7 are positive cases. In 6,7 \t is considered as white space but not space and thus should not remove multiple \t. Case 8 and 9 checks invalid input type. The function expects only string input type. In 10 it only multiple spaces are given as an input and it expects only one space as an output string.

Thus, the test suite tests empty string, non empty string, null input, multiple continuous spaces, whitespaces, single spaces, no spaces and different input types.;