IT314 Software Engineering

Lab 8: Functional Testing (Black-Box)

Name: Rit Rajendra Trambadia **Id: 202201424**

1. **Consider a program for determining the previous date. Its input is triple of day, month and year with the following ranges 1 <= month <= 12, 1 <= day <= 31, 1900 <= year <= 2015.The possible output dates would be previous date or invalid date. Design the equivalence class test cases?**

Valid Input Cases

Test Case ID	Input (Day, Month, Year)	Description	Expected Output
TC1	(1, 1, 2000)	Valid Date (Non-leap year, start of the year)	31/12/1999
TC2	(1, 3, 2004)	Valid Date (Leap year, start of March)	29/02/2004
TC3	(1, 3, 2001)	Valid Date (Non-leap year, end of February)	28/02/2001
TC4	(1, 1, 1901)	Valid Date (End of year)	31/12/1900
TC5	(1, 5, 2010)	Valid Date (End of April)	30/04/2010

Invalid Input Cases

TC6	(10, 13, 2005)	Invalid Month (Above valid range)	Invalid Date
TC7	(15, 0, 2010)	Invalid Month (Below valid range)	Invalid Date
TC8	(32, 1, 2000)	Invalid Day (Above valid range for January)	Invalid Date
TC9	(30, 2, 2004)	Invalid Day (Above valid range for February in leap year)	Invalid Date
TC10	(15, 5, 1899)	Invalid Year (Below valid range)	Invalid Date
TC11	(15, 5, 2016)	Invalid Year (Above valid range)	Invalid Date

Boundary Case Test Cases

TC12	(1, 1, 1900)	Boundary Year (Minimum valid year)	Invalid Date
TC13	(1, 1, 2015)	Boundary Year (Maximum valid year)	31/12/2014
TC14	(1, 12, 2015)	Boundary Month (End of valid month)	30/11/2015
TC15	(1, 2, 2000)	Leap Year February (Valid leap year boundary)	29/01/2000
TC16	(1, 3, 2001)	Non-leap Year February (End of February, non-leap year)	28/02/2001
TC17	(30, 4, 2010)	Boundary Day (Last valid day of April)	29/04/2010

Edge Case Test Cases

TC18	(1, 3, 2000)	Leap Year February (End of February, leap year)	29/02/2000
TC19	(1, 3, 1900)	Century year non-leap (End of February, non-leap)	28/02/1900
TC20	(1, 1, 2001)	Start of a new millennium	31/12/2000

2. Programs:

P1. The function linearSearch searches for a value v in an array of integers a. If v appears in the array a, then the function returns the first index i, such that a[i] == v; otherwise, -1 is returned.

Valid Input Cases

Test Case ID	Input (Array a, Value v)	Description	Expected Output
v is in the middle of the array	([1, 2, 3, 4, 5], 3)	Value v is in the middle of the array	2
v is the first element	([1, 2, 3, 4, 5], 1)	Value v is the first element	0
v is the last element	([1, 2, 3, 4, 5], 5)	Value v is the last element	4
v appears multiple times	([1, 1, 1, 1, 1], 1)	All elements in the array are the same as v	0
v is absent from the array	([1, 2, 3, 4, 5], 6)	Value v does not exist in the array	-1
Empty array	([], 3)	Empty array, no elements to search	-1
Single element array, v is present	([10], 10)	Single element array, v is the element	0
Single element array, v is absent	([10], 5)	Single element array, v is not in the array	-1

Invalid Input Cases

Test Case ID	Input (Array a, Value v)	Description	Expected Output
a contains float elements	([1.5, 2.5, 3.5], 2)	Array contains float numbers	Invalid Input
v is a float	([1, 2, 3], 2.5)	Search value v is a float	Invalid Input
a contains non-integer elements	(['a', 'b', 'c'], 1)	Array contains non-integer elements	Invalid Input

Boundary Value Test Cases

Test Case ID	Input (Array a, Value v)	Description	Expected Output
Array of all identical elements, v is present	([0, 0, 0, 0, 0], 0)	Array has repeating zeros, v is zero	0
v is the first element in the array	([100, 200, 300, 400], 100)	Value v is the first element in an array of larger numbers	0
v is the last element in the array	([3, 5, 7, 9, 11, 13], 13)	Value v is the last element in the array	5
Single element array, v is absent	([1], 2)	Single element array, but v is not present.	-1

P2. The function countItem returns the number of times a value v appears in an array of integers a.

Valid Input Cases

Equivalence Class	Input (Array a, Value v)	Description	Expected Output
v appears multiple times	([1, 2, 3, 2, 1], 2)	Value v appears two times in the array.	2
all elements are equal to v	([1, 1, 1, 1, 1], 1)	All elements in the array are the same as v.	5
v does not appear in the array	([1, 2, 3, 4, 5], 6)	Value v is not present in the array.	0
empty array	([], 3)	Array is empty, so v cannot appear.	0
single element array, v is present	([10], 10)	Only one element which is equal to v.	1
single element array, v is absent	([10], 5)	Only one element which is not equal to v.	0
array contains negative and positive values	([-3, -2, -1, 0, 1], 0)	v is zero and appears once.	1

Invalid Input Cases

Equivalence Class	Input (Array a, Value v)	Description	Expected Output
array contains float numbers	([1.5, 2.5, 3.5], 2)	Array contains float numbers, which are invalid.	Invalid Input
search value v is a float	([1, 2, 3], 2.5)	v is a float, which is invalid.	Invalid Input
array contains non-integer elements	(['a', 'b', 'c'], 'a')	Array contains non-integer elements, which are invalid.	Invalid Input

Boundary Value Test Cases

Equivalence Class	Input (Array a, Value v)	Description	Expected Output
array has repeating zeros	([0, 0, 0, 0, 0], 0)	v is zero and appears five times.	5
value v is the first element	([100, 200, 300, 400], 100)	v is the first element in the array.	1

value v is the last element	([3, 5, 7, 9, 11, 13], 13)	v is the last element in the array.	1
value v is negative and found	([-10, -5, 0, 5, 10], -5)	v is negative and appears once.	1

P3. The function binarySearch searches for a value v in an ordered array of integers a. If v appears in the array a, then the function returns an index i, such that a[i] == v; otherwise, -1 is returned.

Valid Input Cases

Equivalence Class	Input (Array a, Value v)	Description	Expected Output
v is the first element	([1, 2, 3, 4, 5], 1)	v is the first element in the ordered array.	0
v is the last element	([1, 2, 3, 4, 5], 5)	v is the last element in the ordered array.	4
v appears in the middle	([1, 2, 3, 4, 5], 3)	v appears in the middle of the ordered array.	2
v does not appear in the array	([1, 2, 3, 4, 5], 6)	v is greater than the largest element in the array.	-1
v is less than the smallest element	([1, 2, 3, 4, 5], 0)	v is less than the smallest element in the array.	-1
empty array	([], 1)	Array is empty, so v cannot appear.	-1
single element array, v is present	([10], 10)	Only one element which is equal to v.	0
single element array, v is absent	([10], 5)	Only one element which is not equal to v.	-1
v is a negative number	([-5, -3, -1, 0, 1], -3)	v is negative and exists in the ordered array.	1
array contains multiple entries of v	([1, 2, 2, 2, 3], 2)	v appears multiple times; first index is returned.	1

Invalid Input Cases

Equivalence Class	Input (Array a, Value v)	Description	Expected Output
array contains float numbers	([1.5, 2.5, 3.5], 2)	Array contains float numbers, which are invalid.	Invalid Input
search value v is a float	([1, 2, 3], 2.5)	v is a float, which is invalid.	Invalid Input
array contains non-integer elements	(['a', 'b', 'c'], 'a')	Array contains non-integer elements, which are invalid.	Invalid Input

Boundary Value Test Cases

Equivalence Class	Input (Array a, Value v)	Description	Expected Output
v is the minimum value in the array	([1, 2, 3, 4, 5], 1)	v is the minimum value and appears in the array.	0
v is the maximum value in the array	([1, 2, 3, 4, 5], 5)	v is the maximum value and appears in the array.	4
v is just less than the maximum value	([1, 2, 3, 4, 5], 4)	v is just less than the maximum value and exists.	3
v is just greater than the minimum value	([1, 2, 3, 4, 5], 2)	v is just greater than the minimum value and exists.	1
v is in the middle of a large array	([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 5)	v is in the middle of a larger ordered array.	4
array contains multiple entries of v	([1, 1, 2, 2, 3], 2)	v appears multiple times; returns the first index.	2

P4. The following problem has been adapted from The Art of Software Testing, by G. Myers (1979). The function triangle takes three integer parameters that are interpreted as the lengths of the sides of a triangle. It returns whether the triangle is equilateral (three lengths equal), isosceles (two lengths equal), scalene (no lengths equal), or invalid (impossible lengths).

Valid Input Cases

Equivalence Class	Input (Sides a, b, c)	Description	Expected Output
All sides are equal	(3, 3, 3)	An equilateral triangle (all sides equal).	EQUILATERAL
Two sides are equal	(3, 3, 2)	An isosceles triangle (two sides equal).	ISOSCELES
Two sides are equal but reversed	(2, 3, 3)	An isosceles triangle (two sides equal).	ISOSCELES
All sides are different	(3, 4, 5)	A scalene triangle (no sides equal).	SCALENE
All sides are different (larger values)	(5, 6, 7)	A scalene triangle (no sides equal).	SCALENE
Minimal valid triangle (a + b > c)	(1, 1, 1)	An equilateral triangle with minimal values.	EQUILATERAL
Isosceles triangle with minimal values	(1, 1, 2)	Not a valid triangle (impossible lengths).	INVALID

Invalid Input Cases

Equivalence Class	Input (Sides a, b, c)	Description	Expected Output
One side greater than the sum of the others	(5, 2, 2)	Not a valid triangle (5 >= 2 + 2).	INVALID
Two sides equal but longer than the third	(3, 3, 7)	Not a valid triangle (3 + 3 <= 7).	INVALID
All sides are zero	(0, 0, 0)	Not a valid triangle (zero-length sides).	INVALID
Negative lengths	(-1, 2, 3)	Not a valid triangle (negative length).	INVALID
Negative and positive lengths	(-1, -1, 2)	Not a valid triangle (negative length).	INVALID
One side is negative	(3, 4, -5)	Not a valid triangle (negative length).	INVALID

Boundary Value Test Cases

Equivalence Class	Input (Sides a, b, c)	Description	Expected Output
Smallest valid triangle	(1, 1, 1)	An equilateral triangle (all sides equal).	EQUILATERAL
Smallest invalid triangle	(1, 1, 2)	Not a valid triangle (1 + 1 <= 2).	INVALID
One side is zero	(0, 1, 1)	Not a valid triangle (zero-length side).	INVALID
Minimum positive side with zero	(1, 1, 0)	Not a valid triangle (zero-length side).	INVALID
Two equal sides with one greater	(3, 3, 6)	Not a valid triangle (3 + 3 <= 6).	INVALID
Largest valid triangle	(10, 10, 10)	An equilateral triangle (all sides equal).	EQUILATERAL
Largest invalid triangle (long side)	(10, 10, 21)	Not a valid triangle (10 + 10 <= 21).	INVALID

P5. The function prefix (String s1, String s2) returns whether or not the string s1 is a prefix of string s2 (you may assume that neither s1 nor s2 is null).

Valid Input Cases

Equivalence Class	Input (String s1, String s2)	Description	Expected Output
s1 is an empty string	("", "abc")	s1 is empty, which is considered a prefix of any string.	True
s1 is equal to s2	("abc", "abc")	s1 is equal to s2, so it is a prefix.	True
s1 is a proper prefix of s2	("ab", "abc")	s1 is a proper prefix of s2.	True
s1 is longer than s2	("abcd", "abc")	s1 is longer than s2; cannot be a prefix.	False
s1 is not a prefix of s2	("ac", "abc")	s1 is not a prefix of s2.	False
s1 is a single character prefix	("a", "abc")	s1 is a single character that is a prefix of s2.	True
s1 is the first character of s2	("a", "abc")	s1 is the first character of s2.	True
s1 is a substring at the end	("bc", "abc")	s1 is a substring but not a prefix.	False
s1 is a single character not a prefix	("b", "abc")	s1 is a single character that is not a prefix of s2.	False

Invalid Input Cases

Equivalence Class	Input (String s1, String s2)	Description	Expected Output
s1 contains special characters	("@!", "abc")	s1 contains special characters.	False
s1 contains spaces	("ab c", "ab cdef")	s1 contains spaces, which are valid in prefix check.	True
s2 contains special characters	("abc", "@!#")	s2 contains special characters.	False
s2 contains spaces	("abc", "ab cdef")	s2 contains spaces.	False
s1 is numeric string	("123", "123abc")	s1 is a numeric string and is a prefix of s2.	True
s2 is numeric string	("abc", "123")	s2 is numeric and not a prefix.	False

Boundary Value Test Cases

Equivalence Class	Input (String s1, String s2)	Description	Expected Output
s1 is empty and s2 is empty	("", "")	Both strings are empty; s1 is a prefix of s2.	True
s1 is empty and s2 is non-empty	("", "abc")	s1 is empty; it is a prefix of any non-empty s2.	True
s1 is non-empty and s2 is empty	("abc", "")	Non-empty s1 cannot be a prefix of empty s2.	False
s1 is a single character and s2 is empty	("a", "")	Non-empty s1 cannot be a prefix of empty s2.	False
s1 is a long string and s2 is short	("abcdefghijkl", "abc")	s1 is longer than s2; cannot be a prefix.	False
s1 is a prefix that exactly matches s2 length	("abc", "abc")	s1 is exactly equal to s2.	True
s1 is longer than s2 but matches at the start	("abcd", "abc")	s1 cannot be a prefix of s2 as it is longer.	False

P6: Consider again the triangle classification program (P4) with a slightly different specification: The program reads floating values from the standard input. The three values A, B, and C are interpreted as representing the lengths of the sides of a triangle. The program then prints a message to the standard output that states whether the triangle, if it can be formed, is scalene, isosceles, equilateral, or right angled. Determine the following for the above program:

a) Identify the Equivalence Classes

Equivalence Classes for Triangle Types:

- **Equilateral Triangle**: $a = b = c$
- **Isosceles Triangle**: $a = b \neq c$ or $a = c \neq b$ or $b = c \neq a$
- **Scalene Triangle**: $a \neq b \neq c$
- **Right-Angled Triangle**: $a^2+b^2=c^2$ (assuming C isthe longest side)
- **Invalid Triangle**: $a + b \leq c$ or $a + c \leq b$ or $b + c \leq a$
- **Non-Triangle (Non-positive lengths)**: $a \leq 0$ or $b \leq 0$ or $c \leq 0$

b) Identify Test Cases to Cover the Identified Equivalence Classes

Test Case	Input Values (a, b, c)	Expected Outcome	Equivalence Class Covered
Test Case 1	(3.0, 3.0, 3.0)	EQUILATERAL	**Equilateral Triangle**
Test Case 2	(5.0, 5.0, 3.0)	ISOSCELES	**Isosceles Triangle**
Test Case 3	(4.0, 5.0, 6.0)	SCALENE	**Scalene Triangle**
Test Case 4	(3.0, 4.0, 5.0)	RIGHT-ANGLED	**Right-Angled Triangle**
Test Case 5	(1.0, 2.0, 3.0)	INVALID	**Invalid Triangle**
Test Case 6	(1.0, 2.0, 0.0)	NON-TRIANGLE	**Non-Triangle (Non-positive lengths)**
Test Case 7	**(0.0, 5.0, 5.0)**	NON-TRIANGLE	**Non-Triangle (Non-positive lengths)**

c) Boundary Condition for Scalene Triangle (a ≠ b ≠ c)

Test Case	Input Values (a, b, c)	Expected Outcome
Test Case1	(3.0, 4.0, 5.0)	**SCALENE**
Test Case2	(2.0, 3.0, 4.0)	**SCALENE**
Test Case3	(3.0, 4.0, 7.0)	**INVALID**

d) Boundary Condition for Isosceles Triangle (a = c)

Test Case	Input Values (a, b, c)	Expected Outcome
Test Case1	(5.0, 5.0, 3.0)	**ISOSCELES**
Test Case2	(5.0, 3.0, 5.0)	**ISOSCELES**
Test Case3	(5.0, 5.0, 5.0)	**EQUILATERAL**
Test Case4	(0.0, 5.0, 0.0)	**NON-TRIANGLE**

e) Boundary Condition for Equilateral Triangle (a = b = c)

Test Case	Input Values (a, b, c)	Expected Outcome
Test Case1	(3.0, 3.0, 3.0)	**EQUILATERAL**
Test Case2	(0.0, 0.0, 0.0)	**NON-TRIANGLE**
Test Case3	(5.0, 5.0, 5.0)	**EQUILATERAL**

f) Boundary Condition for Right-Angled Triangle ($a^2 + b^2 = c^2$)

Test Case	Input Values (a, b, c)	Expected Outcome
Test Case1	(3.0, 4.0, 5.0)	**RIGHT-ANGLED**
Test Case2	(5.0, 12.0, 13.0)	**RIGHT-ANGLED**
Test Case3	(8.0, 15.0, 17.0)	**RIGHT-ANGLED**

g) Non-Triangle Case

Test Case	Input Values (a, b, c)	Expected Outcome
Test Case1	(1.0, 2.0, 3.0)	**INVALID**
Test Case2	(3.0, 1.0, 1.0)	**INVALID**
Test Case3	(5.0, 5.0, 10.0)	**INVALID**
Test Case4	(7.0, 3.0, 4.0)	**INVALID**

h) Non-Positive Input

Test Case	Input Values (a, b, c)	Expected Outcome
Test Case1	(-1.0, 2.0, 3.0)	**NON-TRIANGLE**
Test Case2	(0.0, 5.0, 5.0)	**NON-TRIANGLE**
Test Case3	(5.0, 0.0, 5.0)	**NON-TRIANGLE**
Test Case4	(5.0, 5.0, -1.0)	**NON-TRIANGLE**
Test Case5	(0.0, 0.0, 0.0)	**NON-TRIANGLE**