IT-314 Software Engineering

Lab8-Functional Testing (black Boxing)

202201076-Vidhi Dhanani

Q.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?

Write a set of test cases (i.e., test suite) – specific set of data – to properly test the programs.
Your test suite should include both correct and incorrect inputs.

Equivalence Class	**Description**	**Valid or Invalid**
E1	1<=day<=31	valid
E2	day<1	invalid
E3	day>31	Invalid
E4	1<=month<=12	Valid
e5	month<1	invalid
E6	month>12	Invalid
E7	1900<=year<=2015	Valid
E8	year<1900	Invalid
E9	year>2015	Invalid

1. Enlist which set of test cases have been identified using Equivalence Partitioning and Boundary Value Analysis separately.

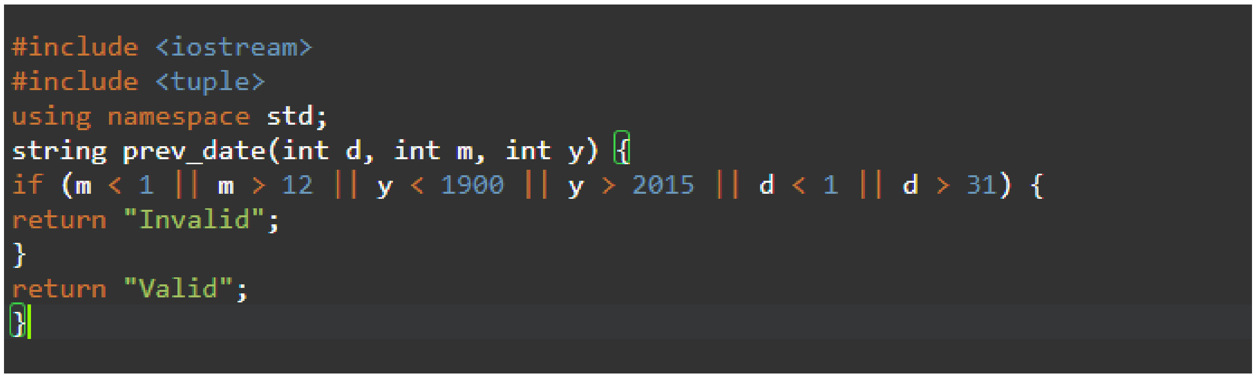
Equivalence test-case Analysis

No	**date**	**Month**	**year**	**Covered Equivalence class**	**Expected output**
1	1	1	1901	E1, E4, E7	Invalid date
2	30	12	2014	E1, E4, E7	29-12-2014
3	5	5	2002	E1, E4, E7	4-5-2002
4	29	2	2000	E1, E4, E7	28-2-2000
5	15	3	2003	E1, E4, E7	14-3-2003
6	0	11	2005	E2, E4, E7	Invalid date
7	32	1	2010	E3, E4, E7	Invalid date
8	31	6	2011	E1, E4, E7	30-6-2011
9	20	13	2006	E6, E1, E7	Invalid date
10	5	5	1895	E1, E4, E8	Invalid date
11	5	5	2017	E1, E4, E9	Invalid date
12	29	2	2016	E1, E4, E7	28-2-2016

Boundary test case analysis:

No	**Date**	**Month**	**Year**	**Covered Equivalence class**	**Expected output**
1	1	1	1900	E1,E4,E7	Invalid
2	31	12	2015	E1,E4,E7	30-12-2015
3	1	12	1900	E1,E4,E7	30-11-1900
4	1	1	2015	E1,E4,E7	31-12-2014
5	29	2	2000	E1,E4,E7	28-2-2000
6	28	2	2001	E1,E4,E7	27-2-2001
7	30	11	2014	E1,E4,E7	29-11-2014
8	31	1	2014	E1,E4,E7	30-01-2000-
9	1	2	2001	E1,E4,E7	31-1-2001
10	29	2	1901	E1,E4,E7	28-2-1901

2. Modify your programs such that it runs, and then execute your test suites on the program.While executing your input data in a program, check whether the identified expected outcome (mentioned by you) is correct or not.
The solution of each problem must be given in the format as follows:
Tester Action and Input Data Expected Outcome
Equivalence Partitioning
a, b, c An Error message
a-1, b, c Yes
Boundary Value Analysis
a, b, c-1 Yes



```
#include <iostream>
#include <tuple>
using namespace std;
string prev_date(int d, int m, int y) {
if (m < 1 || m > 12 || y < 1900 || y > 2015 || d < 1 || d > 31) {
return "Invalid";
}
return "Valid";
}
```

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.

● Equivalence Partitioning (EP):
Equivalence Class Description:
● E1: The array is empty.
● E2: The value v is present in the array.
● E3: The value v is not present in the array.
● E4: The array contains only one element which is equal to v.
● E5: The array contains only one element which is not equal to v.

Test Cases:

Number	Input Data(Array a, Value v)	Expected Outcome	Covered Equivalence Class
1.	a = [ ], v = 5	-1	E1
2.	a = [1,2,3,4,5], v = 5	4	E2
3.	a = [1,2,3,4,6], v = 5	-1	E3
4.	a = [5], v = 5	0	E4
5.	a = [4], v = 5	-1	E5

Boundary Value Analysis:

● The array size is at its minimum size, either empty or contains just one element.
● The value is at the start or end of the array.
● The value is not present, but close to elements in the array.

Test Case	Input Data (Array a,Value v)	Expected Outcome	Boundary Condition
1.	a = [5], v = 5	0	Single element array, value present
2.	a = [5], v = 6	-1	Single element array, value absent
3.	a = [1,2,3,4,5], v = 1	0	Value is at the start of the array
4.	v = 5	4	Value is at the end of

			the array
5.	a = [1,2,3,4,5], v = 6	-1	Value absent but close to elements in the array

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

● **By Equivalence Class:-**

1. Array contains multiple occurrences of the value.
2. Array does not contain the value.
3. Empty array.
4. Single element array (element is the value).
5. Single element array (element is not the value).

Test-Case	**Expected Outcome**	**Equivalence Class**
v=1, a[ ] =[1,2,1]	2	1
v=1, a[ ] =[2,3,4]	0	2
v=1, a[ ] =[1]	0	3
v=2, a[ ] =[2]	1	4
v=2, a[ ] = [3]	0	5

● **Boundary Value Analysis :**

Test-Case	**Expected Outcome**
v=1, a[ ] =[1,2,3]	1
v=1, a[ ] =[2,3,1]	1

v=1, a[ ] =[ ]	0
v=2, a[ ] =[2]	1
v=2, a[ ] = [3]	0

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. Assumption: the elements in the array a are sorted in non-decreasing order.

Equivalence Class Description:

- E1: The array is empty.
- E2: The value v is present in the array.
- E3: The value v is not present in the array.
- E4: The array contains only one element which is equal to v.
- E5: The array contains only one element which is not equal to v.

Test Case	**Input Data (Array a,Value v)**	**Expected Outcome**	**Equivalence Class**
1.	a = [ ], v = 5	-1	E1
2.	a = [1,2,3,4,5], v = 5	4	E2
3.	a = [1,2,3,4,6], v = 5	-1	E3
4.	a = [5], v = 5	0	E4
5.	a = [4], v = 5	-1	E5

Boundary Conditions:

● The array size is at its minimum size, either empty or contains just one element.

● The value is at the start, middle, or end of the array.

● The value is not present but close to elements in the array.

Test Case	Input Data (Array a,Value v)	Expected Outcome	Boundary Condition
1.	a = [5], v = 5	0	Single element array, value present
2.	a = [5], v = 6	-1	Single element array, value absent
3.	a = [1,2,3,4,5], v = 1	0	Value is at the start of the array
4.	a = [1,2,3,4,5], v = 3	2	Value is in middle of array
5.	a = [1,2,3,4,5], v = 5	4	Value is at end of array
6.	A = [1,2,3,4,5], v = 6	-1	Value absent but close to elements in array

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).

● By Equivalence Class:

1. Valid equilateral triangle:
2. Valid isosceles triangle:
3. Valid scalene triangle:
4. Invalid triangle (sum of any two sides must be greater than the third):
5. Negative lengths:
6. Zero lengths:

● **Test-Case:**

Test-Case	**Expected Outcomes**
a=3 ,b=3,c=3	Equilateral
a=4,b=4,c=5	Isosceles
a=3,b=4,c=5	Scalene
a=1,b=2,c=3	InvaliD
a=-1,b=3,c=4	Invalid
a=0,b=3,c=4	Invalid

● **Boundary Value Analysis :**

Test-Case	**Expected Outcomes**
a = 2, b = 2, c = 2	Invalid
a = 1, b = 1, c = 2	Equilateral
a = -1, b = 1, c = 1	Invalid
a = 0, b = 1, c = 1	Invalid
a = 1, b = 1, c = 1	Equilateral

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).

Equivalence Partitioning (EP):

Equivalence Class Description:

- E1: s1 is longer than s2 (impossible to be a prefix).
- E2: s1 is a valid prefix of s2.
- E3: s1 is not a prefix of s2.
- E4: s1 is an empty string (edge case).
- E5: s2 is an empty string (edge case).

Equivalence Class Test Cases:

Test Case	**Input Data (s1, s2)**	**Expected Outcome**	**Covered Equivalence Class**
TC1	"abcdef", "abc"	false	E1
TC2	"abc", "abcdef"	true	E2
TC3	"xyz", "abcdef"	false	E3
TC4	" ", "abcdef"	true	E4
TC5	"abc", ""	false	E5

Boundary Value Analysis (BVA):

Boundary Conditions:

- Length of s1 is greater than the length of s2.
- s1 is an empty string, s2 is a non-empty string.
- s2 is an empty string, s1 is non-empty.
- s1 equals s2.

Boundary Value Test Cases:

Test Case	**Input Data (s1, s2)**	**Expected Outcome**	**Boundary Condition**
TC1	“a”,” ”	false	S2 is empty
TC2	"abcdef", "abcdef"	true	s1 equals s2
TC3	"abc", "abc"	true	Shorter but equal strings
TC4	" ", " "	true	Both strings are empty

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 for the system
b) Identify test cases to cover the identified equivalence classes. Also, explicitly mention which test case would cover which equivalence class. (Hint: you must need to be ensure that the identified set of test cases cover all identified equivalence classes)
c) For the boundary condition A + B > C case (scalene triangle), identify test cases to verify the boundary.
d) For the boundary condition A = C case (isosceles triangle), identify test cases to verify the boundary.

e) For the boundary condition A = B = C case (equilateral triangle), identify test cases to verify the boundary.
f) For the boundary condition A2 + B2 = C2 case (right-angle triangle), identify test cases to verify the boundary.
g) For the non-triangle case, identify test cases to explore the boundary.
h) For non-positive input, identify test points.

● **By Equivalence Class:**
1. Valid equilateral triangle: All sides are equal.
2. Valid isosceles triangle: Exactly two sides are equal.
3. Valid scalene triangle: All sides are different.
4. Valid right-angled triangle: Follows the Pythagorean theorem.
5. Invalid triangle (non-triangle): Sides do not satisfy triangle inequalities.
6. Invalid input (non-positive values): one or more sides are non-positive.

Test Case	**Output**	**Equivalence Class**
A = 3.0, B = 3.0, C = 3.0	Equilateral	E1
A = 4.0, B = 4.0, C = 5.0	Isosceles	E2
A = 3.0, B = 4.0, C = 5.0	Scalene	E3
A = 3.0, B = 4.0, C = 6.0	Invalid	E5
A = -1.0, B = 2.0, C = 3.0	Invalid	E6
A = 5.0, B = 12.0, C = 13.0	Right-Angled	E4

c) Boundary conditions for A + B > C (Scalene Triangle)

Test Case	Output
A = 1.0, B = 1.0, C = 1.9999	Scalene
A = 2.0, B = 3.0, C = 4.0	Scalene

d) Boundary Conditions for A=C (Isosceles Triangle)

Test Case	Output
A = 3.0, B = 3.0, C = 4.0	Isosceles
A = 2.0, B = 2.0, C = 3.0	Isosceles
A = 2.0, B = 2.0, C = 2.0	Equilateral

E) Boundary Conditions for A = B = C (Equilateral Triangle)

Test-Case	output
A = 2.0, B = 2.0, C = 2.0	Equilateral
A = 1.9999, B = 1.9999, C = 1.9999	Equilateral

f) Boundary Conditions for A2 + B2 = C2 (Right-Angle Triangle)

Test-Case	Output
A = 3.0, B = 4.0, C = 5.0	Right-angled
A = 5.0, B = 12.0, C = 13.0	Right-angled

g) Test Cases for Non-Triangle Case

Test-Case	Output

A = 1.0, B = 2.0, C = 3.0	Invalid
A = 1.0, B = 2.0, C = 2.0	Invalid
A = 1.0, B = 1.0, C = 3.0	Invalid

h) Test Cases for Non-Positive Input

Test-Case	**Output**
A = 0.0, B = 2.0, C = 3.0	Invalid
A = -1.0, B = -2.0, C = 3.0	Invalid