IT - 314

Software Engineering

Lab - 8

Name - Hitarth Bhatt

Roll No. - 202201024

Question 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?

Equivalence Classes:

No.	Equivalence Classes	Validity
EC1	Month <1	Invalid
EC2	1 ≤ Month ≤ 12	Valid
EC3	Month > 12	Invalid
EC4	Day < 1	Invalid
EC5	1 ≤ Day ≤ 31	Valid
EC6	Day > 31	Invalid
EC7	Year < 1900	Invalid
EC8	1900 ≤ Year ≤ 2015	Valid
EC9	Year > 2015	Invalid

Test Cases:

Test Case No.	Input (Day,Month ,Year)	Class Addressed	Expected Output	Validity	Remarks
1	15,6,2010	EC2,EC5,EC8	14,6,2010	Valid	-
2	12,0,2004	EC1,EC5,EC8	Error Message	Invalid	Invalid Month
3	15, 13, 2005	EC3,EC5,EC8	Error Message	Invalid	Invalid Month
4	1, 1, 1900	EC2,EC5,EC8	31,12,1889	Valid	Boundary Value
5	0,11,2007	EC4,EC2,EC8	Error Message	Invalid	Invalid Day
6	15,9,1889	EC7,EC2,EC5	Error Message	Invalid	Invalid Year
7	32,11,2004	EC6,EC2,EC8	Error Message	Invalid	Invalid Day
8	31,12,2015	EC2,EC5,EC8	30,12,2015	Valid	Boundary Value
9	22,10,2024	EC9,EC2,EC5	Error Message	Invalid	Invalid Year
10	1, 3, 2008	EC2,EC5,EC8	29,2,2008	Valid	Leap Year Transition

Question 2

Part-1

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.

```
int linearSearch(int v,int a[]){
    int i=0;
    while(i<a.length){
        if (a[i]==v)
            return (i);
        i++;
    }
    return (-1);
}
```

Equivalence Class:

No.	Equivalence Classes	Validity
EC1	Value v is not present in array	Valid
EC2	Value v is present in array	Valid
EC3	Array a is empty	Invalid
EC4	Array a contains one or more non integer values	Invalid
EC5	v is non integer value	Invalid
EC6	Array a has all integers value	Valid
EC7	Value v is null	Invalid
EC8	Value v is an integer value	Valid

Test Cases

Test Case	v	a	Classes Covered	Expected Outcome	Remarks
TC1	9	[1, 5, 7, 9]	EC2, EC6, EC8	Index 3	Valid input
TC2	5	[]	EC3, EC8	Error Message	Array a is empty
TC3	Null	[1, 2, 3]	EC6,EC7	Error Message	Value of v is null
TC4	4	[1,5,8,9]	EC1, EC6, EC8	-1	Value v is not in array a
TC5	“3”	[1,3,4]	EC5,EC6	Error Message	Value of v is non integer
TC6	3	[1, 3, 3, 3, 5]	EC2, EC6, EC8	Index 1	Multiple occurrence of value v
TC7	3	[3, 5, “6”, 8]	EC4 , EC8	Error Message	Array Contains Non integer value
TC8	5	[5]	EC2, EC6, EC8	Index 0	Single element array but present in array
TC9	4	[5]	EC1, EC6, EC8	-1	Single element array but not present in array

Part-2

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

```
int countItem(int v,int a[]){
    int count = 0;
    for(int i=0;i<a.length;i++){
        if (a[i]==v)
            count++;
    }
    return (count);
}
```

Equivalence Classes:

No.	Equivalence Classes	Validity
EC1	Value v is not present in array	Valid
EC2	Value v is present in array	Valid
EC3	Array a is empty	Invalid
EC4	Array a contains one or more non integer values	Invalid
EC5	v is non integer value	Invalid
EC6	Array a has all integers value	Valid
EC7	Value v is null	Invalid

Test Cases:

Test Case	v	a	Classes Covered	Expected Outcome	Remarks
TC1	9	[1, 5, 7, 9]	EC2, EC6, EC8	1	Count of value is 1

TC2	5	[]	EC3, EC8	Error Message	Array a is empty
TC3	Null	[1, 2, 3]	EC6,EC7	Error Message	Value of v is null
TC4	4	[1,5,8,9]	EC1, EC6, EC8	0	Value v is not in array a
TC5	“3”	[1,3,4]	EC5,EC6	Error Message	Value of v is non integer
TC6	3	[1, 3, 3, 3, 5]	EC2, EC6, EC8	3	Count of value of v in array a is 3
TC7	3	[3, 5, “6”, 8]	EC4 , EC8	Error Message	Array Contains Non integer value
TC8	5	[5]	EC2, EC6, EC8	1	Single element array but present in array
TC9	4	[5]	EC1, EC6, EC8	0	Single element array but not present in array

Part-3

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.

```
int binarySearch(int v,int a[]){
    int lo,mid,hi;
    lo = 0;
    hi = a.length - 1;
    while(lo<=hi){
        mid = (lo + hi)/2;
        if (v == a[mid])
            return (mid);
        else if (v<a[mid])
            hi = mid - 1;
        else
            lo = mid + 1;
    }
    return (-1);
}
```

Equivalence Classes:

No.	Equivalence Classes	Validity
EC1	Value v is not present in array	Valid
EC2	Value v is present in array	Valid
EC3	Array a is empty	Invalid
EC4	Array a contains one or more non integer values	Invalid
EC5	v is non integer value	Invalid
EC6	Array a has all integers value	Valid
EC7	Value v is null	Invalid
EC8	Value v is an integer value	Valid

Test Cases:

Test Case	v	a	Classes Covered	Expected Outcome	Remarks
TC1	9	[1, 5, 7, 9]	EC2, EC6, EC8	Index 3	Valid input
TC2	5	[]	EC3, EC8	Error Message	Array a is empty
TC3	Null	[3,5,6,9]	EC6,EC7	Error Message	Value of v is null
TC4	4	[1,5,8,9]	EC1, EC6, EC8	-1	Value v is not in array a
TC5	“8”	[1,3,4]	EC5,EC6	Error Message	Value of v is non integer
TC6	3	[1, 3, 3, 3, 5]	EC2, EC6, EC8	Index 1	Multiple occurrence of value v
TC7	3	[1, 2, “5”, 9]	EC4 , EC8	Error Message	Array Contains Non integer value
TC8	5	[5]	EC2, EC6, EC8	Index 0	Single element array but present in array
TC9	4	[5]	EC1, EC6, EC8	-1	Single element array but not present in array

Part-4

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).

```
final int EQUILATERAL = 0;
final int ISOSCELES  = 1;
final int SCALENE = 2;
final int INVALID = 3;
int triangle(int a,int b,int c){
    if (a>=b + c || b>= a + c || c>= a + b)
        return (INVALID);
    if (a == b && b == c)
        return (EQUILATERAL);
    if (a==b || b==c || c==a)
        return (ISOSCELES);
    return (SCALENE);
}
```

Equivalence Classes:

No.	Equivalence Classes	Validity
EC1	One or more side lengths are non-positive integers values	Invalid
EC2	One or more side lengths are null	Invalid
EC3	All side lengths are positive integers	Valid
EC4	One or more side lengths are less than or equal to 0	Invalid
EC5	Triangle inequality is satisfied: a + b > c, b + c > a, a + c > b	Valid
EC6	Triangle inequality is violated	Invalid

EC7	All three sides are equal	Valid
EC8	Exactly two sides are equal	Valid
EC9	No sides are equal	Valid

Test Cases:

Test Case	a	b	c	Classes Covered	Expected Outcome	Remarks
TC1	5	5	5	EC3 , EC5, EC7	EQUILATERAL	All the sides are equal
TC2	5	5	8	EC3 , EC5, EC8	ISOSCELES	Two sides are equal
TC3	5	6	7	EC3 , EC5, EC9	SCALENE	No sides are equal
TC4	1	2	3	EC3, EC6	INVALID	Triangle inequality violated
TC5	5	5	10	EC3, EC6	INVALID	Borderline invalid triangle
TC6	0	5	5	EC4	INVALID	Zero length side
TC7	3	Null	5	EC2	INVALID	Null length side
TC8	7	10	5	EC3 , EC5, EC9	SCALENE	No sides are equal
TC9	1	1	2	EC3, EC6	INVALID	Borderline invalid triangle
TC10	100	100	100	EC3 , EC5, EC7	EQUILATERAL	Large equilateral triangle
TC11	3	3	5	EC3 , EC5, EC8	ISOSCELES	Two sides are equal
TC12	1	“5”	5	EC1	INVALID	Non - integers value of side

Part-5

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).

```
public static boolean prefix(String s1,String s2){
    if (s1.length()>s2.length()){
        return false;
    }
    for(int i=0;i<s1.length();i++){
        if (s1.charAt(i)!=s2.charAt(i)){
            return false;
        }
    }
    return true;
}
```

Equivalence Classes:

No.	Equivalence Classes	Validity
EC1	s1 is an empty string	Valid
EC2	s2 is an empty string	Valid
EC3	s1.length() > s2.length()	Invalid
EC4	s1.length() <= s2.length()	Valid
EC5	All characters of s1 match the corresponding characters of s2	Valid
EC6	At least one character of s1 does not match the corresponding character of s2	Invalid
EC7	s1 equals s2	Valid
EC8	Both s1 and s2 are non-empty but do not match fully	Invalid

Test Cases:

Test Case	s1	s2	Classes Covered	Expected Outcome	Remarks
TC1	“ ”	“rataew”	EC1, EC4, EC5	True	An empty string is a prefix of any string
TC2	“hit”	“hitarth”	EC4, EC5	True	s1 is a valid prefix of s2
TC3	“as”	“tuk”	EC4, EC6	False	s1 is not a prefix of s2
TC4	“hit”	“hit”	EC4, EC5, EC7	True	s1 is equal to s2
TC5	“adi”	“a”	EC3	False	s1 is longer than s2
TC6	“ ”	“ ”	EC1, EC2, EC7	True	Both strings are empty
TC7	“timer”	“harsh”	EC3	False	s1 is longer than s2
TC8	“ha”	“happy”	EC4, EC5	True	s1 is a valid prefix of s2
TC9	“e”	“h”	EC4, EC6	False	Single-character strings do not match

Part - 6

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.

No.	Equivalence Classes	Validity
EC1	One or more side lengths are non-positive floating point values	Invalid
EC2	One or more side lengths are less than or equal to 0	Invalid
EC3	All side lengths are valid positive floating-point values	Valid
EC4	Triangle inequality is satisfied: a + b > c, b + c > a, a + c > b	Invalid
EC5	Triangle inequality is violated	Valid
EC6	All three sides are equal	Invalid
EC7	Exactly two sides are equal	Valid
EC8	No sides are equal	Valid
EC9	Right-angled triangle: A^2 + B^2 = C^2.	Valid

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)

Test Case	a	b	c	Classes Covered	Expected Outcome	Remarks
TC1	"a"	4.0	5.0	EC1	Error/Exception	One side is non-floating-point
TC2	-1.0	3.0	4.0	EC2	INVALID	One side is negative
TC3	0.0	4.0	5.0	EC2	INVALID	One side is zero
TC4	3.0	3.0	3.0	EC3, EC4, EC6	EQUILATERAL	All sides are equal
TC5	3.0	3.0	2.0	EC3, EC4, EC7	ISOSCELES	Two sides are equal
TC6	3.0	4.0	5.0	EC3, EC4, EC8, EC9	RIGHT-ANGLED	$A^2 + B^2 = C^2$
TC7	5.0	4.0	3.0	EC3, EC4, EC8, EC9	RIGHT-ANGLED	Right-angled (reversed order)
TC8	3.0	4.0	8.0	EC3, EC5	INVALID	Violates triangle inequality
TC9	3.0	4.0	7.0	EC3, EC4, EC8	INVALID	Violates triangle inequality
TC10	3.0	3.0	5.9	EC3, EC4, EC7	ISOSCELES	Near boundary for isosceles
TC11	3.0	3.0	6.0	EC3, EC5	INVALID	Violates inequality boundary
TC12	5.0	5.0	7.0	EC3, EC4, EC7	ISOSCELES	Valid isosceles triangle
TC13	1.0	1.0	2.0	EC3, EC4, EC7	INVALID	Just violates inequality boundary

C. **For the boundary condition A + B > C case (scalene triangle), identify test cases to verify the boundary.**

TC9:

- A = 3.0, B = 4.0, C = 7.0
- Reason: This is a valid scalene triangle because A+B=7.0>C A + B = 7.0 > C A+B=7.0>C.

TC8:

- A = 3.0, B = 4.0, C = 8.0
- Reason: This is an invalid triangle because A+B=7.0<C A + B = 7.0 < C A+B=7.0<C, violating the inequality.

D. **For the boundary condition A = C case (isosceles triangle), identify test cases to verify the boundary.**

Test Cases for Isosceles Triangle Boundary Condition

TC10:

- A = 3.0, B = 3.0, C = 5.9
- Reason: This is near the boundary for an isosceles triangle where two sides are almost equal.

TC11:

- A = 3.0, B = 3.0, C = 6.0
- Reason: This is an invalid triangle because the inequality is violated: A+B=CA + B = CA+B=C.

E. For the boundary condition A = B = C case (equilateral triangle), identify test cases to verify the boundary.

Test Case for Equilateral Triangle Boundary Condition

TC4:

- A = 3.0, B = 3.0, C = 3.0
- Reason: All three sides are equal, forming an equilateral triangle.

F. For the boundary condition A^2 + B^2 = C^2 case (right-angle triangle), identify test cases to verify the boundary.

Test Cases for Right-Angled Triangle Boundary Condition

TC6:

- A = 3.0, B = 4.0, C = 5.0
- Reason: This is a right-angled triangle since 3^2 + 4^2 = 5^2

TC7:

- A = 5.0, B = 4.0, C = 3.0
- Reason: This is also a valid right-angled triangle, but with sides in a different order.

G. For the non-triangle case, identify test cases to explore the boundary.

Test Cases for Non-Triangle Boundary Condition

TC8:

- A = 3.0, B = 4.0, C = 8.0

- Reason: This violates the triangle inequality: A+B<CA + B < CA+B<C.

TC13:

- A = 1.0, B = 1.0, C = 2.0
- Reason: This just fails the inequality: A+B=CA + B = CA+B=C, making it invalid.

H. For non-positive input, identify test points.

Test Cases for Non-Positive Inputs

TC2:

- A = -1.0, B = 3.0, C = 4.0
- Reason: One side is negative, making it an invalid input.