Lab - 8 Software Testing
202201429 (Isha Bhanushali)

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 class:

E1 : input month >=1 and <=12 (valid)
E2 : input month < 1 (invalid)
E3 : input month >12 (invalid)
E4 : input day > 31 (invalid)
E5 : input day <1 (invalid)
E6 : input day >=1 and <=28 and month=2 and [year!=leap year] (valid)
E7: input day >=1 and <=29 and month=2 and year=leap year (valid)
E8 : input day >=1 and <=30 and month=4 or 6 or 9 or 11 (valid)
E9: input day >=1 and <=31 and month = 1 or 3 or 5 or 7 or 8 or 10 or 12.(valid)
E10: input day>30 and month=4 or 6 or 9 or 11 (invalid)
E11 : input day>29 and month=2 (invalid)
E12 : input day>28 for month=2 and year=non-leap (invalid)
E13 : input year>=1900 and <=2015 (valid)
E14 : input year >2015 (invalid)
E15 : input year <1990 (invalid)

- ## Boundary Cases :

B1: (1, *, *) → Valid

B2: (12, *, *) → Valid

B3: (0, *, *) → Invalid

B4: (13, *, *) → Invalid

B5: (32, 1, *) → Invalid

B6: (0, 1, *) → Invalid

B7: (28, 2, 2015) → Valid

B8: (29, 2, 2016) → Valid

B9: (30, 4, *) → Valid

B10: (31, 1, *) → Valid

B11: (31, 4, *) → Invalid

B12: (30, 2, 2015) → Invalid

B13: (29, 2, 2015) → Invalid

B14: (1, 1, 1900) → Valid

B15: (1, 1, 2015) → Valid

B16: (1, 1, 2016) → Invalid

B17: (1, 1, 1899) → Invalid

Test cases:

1. **T1:**
   - Input: (1, 1, 2000)
   - Expected Result: Valid
   - Covered Cases: E1, E9, B1
2. **T2:**
   - Input: (12, 12, 2015)
   - Expected Result: Valid
   - Covered Cases: E1, E9, E13, B2, B15
3. **T3:**
   - Input: (0, 5, 2005)
   - Expected Result: Invalid
   - Covered Cases: E2, B3
4. **T4:**
   - Input: (13, 6, 2010)
   - Expected Result: Invalid
   - Covered Cases: E3, B4
5. **T5:**
   - Input: (32, 1, 2002)
   - Expected Result: Invalid
   - Covered Cases: E4, B5
6. **T6:**

    - Input: (0, 1, 1995)
    - Expected Result: Invalid
    - Covered Cases: E2, B6
7. **T7:**
    - Input: (28, 2, 2015)
    - Expected Result: Valid
    - Covered Cases: E6, B7
8. **T8:**
    - Input: (29, 2, 2016)
    - Expected Result: Valid
    - Covered Cases: E7, B8
9. **T9:**
    - Input: (30, 4, 2003)
    - Expected Result: Valid
    - Covered Cases: E8, B9
10. **T10:**
    - Input: (31, 1, 2005)
    - Expected Result: Valid
    - Covered Cases: E9, B10
11. **T11:**
    - Input: (31, 4, 2012)
    - Expected Result: Invalid
    - Covered Cases: E10, B11
12. **T12:**
    - Input: (30, 2, 2015)
    - Expected Result: Invalid
    - Covered Cases: E11, B12
13. **T13:**
    - Input: (29, 2, 2015)
    - Expected Result: Invalid
    - Covered Cases: E12, B13
14. **T14:**
    - Input: (1, 1, 1900)
    - Expected Result: Valid
    - Covered Cases: E1, E9, E13, B14
15. **T15:**
    - Input: (1, 1, 2016)
    - Expected Result: Invalid
    - Covered Cases: E14, B16
16. **T16:**
    - Input: (1, 1, 1899)
    - Expected Result: Invalid
    - Covered Cases: E15, B17

Question 2:

For p1:

Equivalence Classes

1. **E1**: Value v is an integer and exists in the array a (valid).
2. **E2**: Value v is an integer and does not exist in the array a (invalid).
3. **E3**: Value v is not an integer (invalid).
4. **E4**: Array a is empty (invalid).
5. **E5**: Array contains only one element (valid/invalid depending on v).
6. **E6**: Array contains duplicates of the value v (valid).

Boundary Cases

1. **B1**: v is the first element of a (valid).
2. **B2**: v is the last element of a (valid).
3. **B3**: v is in the middle of a (valid).
4. **B4**: v is not in a but is less than all elements (invalid).
5. **B5**: v is not in a but is greater than all elements (invalid).
6. **B6**: a contains one element which is equal to v (valid).
7. **B7**: a contains one element which is not equal to v (invalid).

Test cases:

Test 1:

- Input: (5, [1, 2, 3, 4, 5])
- Expected Output: 4
- Covered Classes: E1, B2

Test 2:

- Input: (10, [1, 2, 3, 4, 5])
- Expected Output: -1
- Covered Classes: E2, B4

Test 3:

- Input: ("five", [1, 2, 3, 4, 5])
- Expected Output: -1
- Covered Classes: E3

Test 4:

- Input: (1.5, [1, 2, 3, 4, 5])
- Expected Output: -1
- Covered Classes: E3

Test 5:

- Input: (3, [])
- Expected Output: -1
- Covered Classes: E4

Test 6:

- Input: (3, [3])
- Expected Output: 0
- Covered Classes: E1, E5, B6

Test 7:

- Input: (2, [2, 2, 3, 4, 5])
- Expected Output: 0
- Covered Classes: E1, E6, B1

Test 8:

- Input: (6, [1, 2, 3, 4, 5])
- Expected Output: -1
- Covered Classes: E2, B5

Test 9:

- Input: (1, [5])
- Expected Output: -1
- Covered Classes: E2, E5, B7

Test 10:

- Input: (5, [5])
- Expected Output: 0
- Covered Classes: E1, E5, B6

Test 11:

- Input: ("three", [1, 2, 3, 4, 5])
- Expected Output: -1
- Covered Classes: E3

For p2:

Equivalence Classes

.

E1: Value v is an integer and exists in the array a (valid).

E2: Value v is an integer and does not exist in the array a (invalid).

E3: Value v is not an integer (invalid).

E4: Array a is empty (invalid).

E5: Array contains only one element, which may or may not be equal to v (valid/invalid depending on v).

E6: Array contains duplicates of the value v (valid).

Boundary Cases

B1: v is the first element of the array a (valid).

B2: v is the last element of the array a (valid).

B3: v is in the middle of the array a (valid).

B4: v is not in the array but is less than all elements in a (invalid).

B5: v is not in the array but is greater than all elements in a (invalid).

B6: Array a contains only one element which is equal to v (valid).

B7: Array a contains only one element which is not equal to v (invalid).

Test cases:

Test 1:

- Input: `(5, [1, 2, 3, 4, 5])`
- Expected Output: `1`
- Covered Classes: **E1**, **B2**

2. **Test 2**:
   - Input: `(10, [1, 2, 3, 4, 5])`

  - Expected Output: `0`
  - Covered Classes: **E2**, **B5**
3. **Test 3**:
  - Input: `("five", [1, 2, 3, 4, 5])`
  - Expected Output: `0`
  - Covered Classes: **E3**
4. **Test 4**:
  - Input: `(3, [])`
  - Expected Output: `0`
  - Covered Classes: **E4**
5. **Test 5**:
  - Input: `(3, [3])`
  - Expected Output: `1`
  - Covered Classes: **E1**, **E5**, **B6**
6. **Test 6**:
  - Input: `(2, [2, 2, 3, 4, 5])`
  - Expected Output: `2`
  - Covered Classes: **E1**, **E6**, **B1**
7. **Test 7**:
  - Input: `(6, [1, 2, 3, 4, 5])`
  - Expected Output: `0`
  - Covered Classes: **E2**, **B5**
8. **Test 8**:
  - Input: `(1, [1])`
  - Expected Output: `1`
  - Covered Classes: **E1**, **E5**, **B6**
9. **Test 9**:
  - Input: `(5, [5, 5, 5])`
  - Expected Output: `3`
  - Covered Classes: **E1**, **E6**
10. **Test 10**:
  - Input: `(4, [3])`
  - Expected Output: `0`
  - Covered Classes: **E2**, **E5**, **B7**

For p3:

Equivalence Classes

E1: Value v exists in the array a (valid)
E2: Value v does not exist in the array a (invalid)
E3: Array a is empty (invalid).
E4: Value v is less than all elements in the array a (invalid).
E5: Value v is greater than all elements in the array a (invalid).
E6: Array a contains only one element, which may or may not be equal to v (valid/invalid depending on v).

Boundary Cases

B1: v is the first element of the array a (valid).

B2: v is the last element of the array a (valid).

B3: v is in the middle of the array a (valid).

B4: v is less than all elements of a (invalid).

B5: v is greater than all elements of a (invalid).

B6: Array a contains only one element which is equal to v (valid).

B7: Array a contains only one element which is not equal to v (invalid).

Test cases:

Test 1:

- Input: `(5, [1, 2, 3, 4, 5])`
- Expected Output: 4 (index of 5)
- Covered Classes: **E1**, **B2**

Test 2:

- Input: `(10, [1, 2, 3, 4, 5])`
- Expected Output: `-1` (not found)
- Covered Classes: **E2**, **B5**

Test 3:

- Input: `(1, [1, 2, 3, 4, 5])`
- Expected Output: `0` (index of 1)
- Covered Classes: **E1**, **B1**

Test 4:

- Input: `(3, [1, 2, 3, 4, 5])`
- Expected Output: `2` (index of 3)
- Covered Classes: **E1**, **B3**

Test 5:

- Input: `(0, [1, 2, 3, 4, 5])`
- Expected Output: `-1` (not found)
- Covered Classes: **E4**, **B4**

Test 6:

- Input: `(5, [])`
- Expected Output: `-1` (not found)
- Covered Classes: **E3**

Test 7:

- Input: `(3, [3])`
- Expected Output: `0` (index of 3)
- Covered Classes: **E1**, **E6**, **B6**

Test 8:

- Input: `(4, [3])`
- Expected Output: `-1` (not found)
- Covered Classes: **E2**, **E6**, **B7**

Test 9:

- Input: `(5, [5])`
- Expected Output: `0` (index of 5)
- Covered Classes: **E1**, **E6**, **B6**

Test 10:

- Input: `(0, [1])`
- Expected Output: `-1` (not found)

- Covered Classes: **E4**, **E6**, **B4**

For p4:

Equivalence Classes

E1: All three sides are equal (valid, equilateral).
E2: Two sides are equal and one is different (valid, isosceles).
E3: All three sides are different (valid, scalene).
E4: One side is greater than or equal to the sum of the other two sides (invalid)
E5: At least one side is non-positive (invalid).
E6: Not all sides are integers (invalid)
E7: Input is empty (invalid).

Boundary Cases

B1: All sides are positive integers and equal (valid, equilateral).

B2: Two sides are equal, and the third side is the smallest possible positive integer (valid, isosceles).

B3: Two sides are equal, and the third side is just enough to form a triangle (valid, isosceles).

B4: One side is equal to the sum of the other two sides (invalid).
B5: One side is greater than the sum of the other two sides (invalid).

B6: One side is zero or negative (invalid).

B7: Input is empty (invalid).

B8: One or more sides are not integers (invalid).

Test cases:

Test 1:

- Input: `(5, 5, 5)`

- Expected Output: `0` (equilateral)
- Covered Classes: **E1**, **B1**

Test 2:

- Input: `(5, 5, 3)`
- Expected Output: `1` (isosceles)
- Covered Classes: **E2**, **B2**

Test 3:

- Input: `(5, 4, 3)`
- Expected Output: `2` (scalene)
- Covered Classes: **E3**, **B3**

Test 4:

- Input: `(1, 2, 3)`
- Expected Output: `3` (invalid)
- Covered Classes: **E4**, **B4**

Test 5:

- Input: `(1, 1, 2)`
- Expected Output: `3` (invalid)
- Covered Classes: **E4**, **B4**

Test 6:

- Input: `(0, 1, 1)`
- Expected Output: `3` (invalid)
- Covered Classes: **E5**, **B6**

Test 7:

- Input: `(-1, 1, 1)`
- Expected Output: `3` (invalid)
- Covered Classes: **E5**, **B6**

Test 8:

- Input: `("five", 1, 1)`
- Expected Output: `3` (invalid)
- Covered Classes: **E6**, **B8**

Test 9:

- Input: `(5, 5, "three")`
- Expected Output: 3 (invalid)
- Covered Classes: **E6**, **B8**

Test 10:

- Input: `()`
- Expected Output: 3 (invalid)
- Covered Classes: **E7**

Test 11:

- Input: `(2, 2, 4)`
- Expected Output: 3 (invalid)
- Covered Classes: **E4**, **B5**

For p5:

Equivalence Classes

E1: s1 is a valid prefix of s2 (valid).

E2: s1 is longer than s2 (invalid).

E3: s1 is not a prefix of s2 but has the same starting characters (invalid).

E4: s1 is an empty string and s2 is non-empty (valid).

E5: Both s1 and s2 are empty strings (valid).

E6: s1 and s2 are identical (valid).

Boundary Cases

B1: s1 is exactly the same as s2 (valid).

B2: `s1` is an empty string (valid).

B3: `s1` is a non-empty string and `s2` is an empty string (invalid).

B4: `s1` is a single character, and `s2` is a longer string (valid if `s2` starts with that character).

B5: `s1` is a substring that matches the beginning of `s2` but is not a prefix (invalid).

Test cases:

Test 1:

- Input: `("pre", "prefix")`
- Expected Output: `true`
- Covered Classes: **E1**, **B1**

Test 2:

- Input: `("longprefix", "prefix")`
- Expected Output: `false`
- Covered Classes: **E2**, **B3**

Test 3:

- Input: `("pre", "postfix")`
- Expected Output: `false`
- Covered Classes: **E3**

Test 4:

- Input: `("", "nonempty")`
- Expected Output: `true`
- Covered Classes: **E4**, **B2**

Test 5:

- Input: `("", "")`
- Expected Output: `true`
- Covered Classes: **E5**, **B2**

Test 6:

- Input: `("same", "same")`
- Expected Output: `true`
- Covered Classes: **E6**, **B1**

Test 7:

- Input: `("s", "")`
- Expected Output: `false`
- Covered Classes: **E2**, **B3**

Test 8:

- Input: `("a", "apple")`
- Expected Output: `true`
- Covered Classes: **E1**, **B4**

Test 9:

- Input: `("not", "notebook")`
- Expected Output: `true`
- Covered Classes: **E1**

Test 10:

- Input: `("book", "notebook")`
- Expected Output: `false`
- Covered Classes: **E3**, **B5**

For P-6

a) Identify the equivalence classes for the system

E1: All three sides are equal (valid, equilateral).
E2: Two sides are equal and one is different (valid, isosceles).
E3: All three sides are different (valid, scalene).
E4: The triangle satisfies the Pythagorean theorem (valid, right-angled).
E5: One side is greater than or equal to the sum of the other two sides (invalid, cannot form a triangle).
E6: At least one side is non-positive (invalid, cannot form a triangle).
E7: At least one side is negative (invalid, cannot form a triangle).

E8: At least one side is not a valid number (invalid, includes empty input, characters, or strings).
E9: The sides do not form a triangle (valid but can't classify).

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 1:

- **Input**: `(5.0, 5.0, 5.0)`
- **Expected Output**: `Equilateral`
- **Covered Equivalence Classes**: **E1**

Test Case 2:

- **Input**: `(5.0, 5.0, 3.0)`
- **Expected Output**: `Isosceles`
- **Covered Equivalence Classes**: **E2**

Test Case 3:

- **Input**: `(5.0, 4.0, 3.0)`
- **Expected Output**: `Scalene`
- **Covered Equivalence Classes**: **E3**

Test Case 4:

- **Input**: `(3.0, 4.0, 5.0)`
- **Expected Output**: `Right-angled`
- **Covered Equivalence Classes**: **E4**

Test Case 5:

- **Input**: `(1.0, 2.0, 3.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Equivalence Classes**: **E5**

Test Case 6:

- **Input**: `(0.0, 1.0, 1.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Equivalence Classes**: **E6**

Test Case 7:

- **Input**: `(-1.0, 1.0, 1.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Equivalence Classes**: **E7**

Test Case 8:

- **Input**: `(1.0, 1.0, "two")`
- **Expected Output**: `Invalid input`
- **Covered Equivalence Classes**: **E8**

Test Case 9:

- **Input**: `("", 1.0, 1.0)`
- **Expected Output**: `Invalid input`
- **Covered Equivalence Classes**: **E8**

Test Case 10:

- **Input**: `(2.0, 2.0, 4.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Equivalence Classes**: **E5**

Test Case 11:

- **Input**: `(3.0, 0.0, 4.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Equivalence Classes**: **E6**

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

Test Case 1 (Just Valid):

- **Input**: `(3.0, 4.0, 5.0)`
- **Expected Output**: `Scalene`
- **Explanation**: 3+4=7>53 + 4 = 7 > 53+4=7>5 (valid scalene triangle).

Test Case 2 (Boundary Condition):

- **Input**: `(2.0, 2.0, 4.0)`
- **Expected Output**: `Cannot form a triangle`
- **Explanation**: 2+2=42 + 2 = 42+2=4 (invalid as it doesn't satisfy A+B>CA + B > CA+B>C).

Test Case 3 (Just Over Boundary):

- **Input**: `(3.0, 4.0, 6.0)`
- **Expected Output**: `Scalene`
- **Explanation**: 3+4=7>63 + 4 = 7 > 63+4=7>6 (valid scalene triangle).

Test Case 4 (Equal to Boundary):

- **Input**: `(1.0, 2.0, 3.0)`
- **Expected Output**: `Cannot form a triangle`
- **Explanation**: 1+2=31 + 2 = 31+2=3 (invalid as it doesn't satisfy A+B>CA + B > CA+B>C).

Test Case 5 (Negative Edge Case):

- **Input**: `(3.0, 4.0, -1.0)`
- **Expected Output**: `Cannot form a triangle`
- **Explanation**: Negative side length is invalid.

Test Case 6 (Small Values):

- **Input**: `(0.1, 0.2, 0.3)`
- **Expected Output**: `Cannot form a triangle`
- **Explanation**: 0.1+0.2=0.30.1 + 0.2 = 0.30.1+0.2=0.3 (invalid as it doesn't satisfy A+B>CA + B > CA+B>C).

Test Case 7 (Floating Point Precision):

- **Input**: `(0.1, 0.2, 0.30000001)`
- **Expected Output**: `Scalene`
- **Explanation**: 0.1+0.2=0.3>0.300000010.1 + 0.2 = 0.3 > 0.300000010.1+0.2=0.3>0.30000001 (valid scalene triangle).

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

1. Test Case 1 (Valid Isosceles Triangle):
   - Input: `(5.0, 7.0, 5.0)`
   - Expected Output: `Isosceles`
   - Covered Classes: E2
   - Explanation: Two sides are equal (A and C), forming a valid isosceles triangle.
2. Test Case 2 (Valid Isosceles Triangle - Slide Up):
   - Input: `(5.0, 7.1, 5.0)`
   - Expected Output: `Isosceles`
   - Covered Classes: E2
   - Explanation: The condition A=CA = CA=C is still satisfied; thus, it remains a valid isosceles triangle.
3. Test Case 3 (Valid Isosceles Triangle - Slide Down):
   - Input: `(5.0, 6.9, 5.0)`
   - Expected Output: `Isosceles`
   - Covered Classes: E2
   - Explanation: The condition A=CA = CA=C is still satisfied, forming a valid isosceles triangle.
4. Test Case 4 (Invalid Triangle - Sum Violation):
   - Input: `(5.0, 10.0, 5.0)`
   - Expected Output: `Cannot form a triangle`
   - Covered Classes: E5
   - Explanation: The lengths do not satisfy the triangle inequality theorem since 5+5<105 + 5 < 105+5<10.
5. Test Case 5 (Invalid Triangle - Non-Positive Side):
   - Input: `(0.0, 5.0, 0.0)`
   - Expected Output: `Cannot form a triangle`
   - Covered Classes: E6

    - Explanation: At least one side is zero, which cannot form a triangle.
6. Test Case 6 (Invalid Triangle - Negative Side):
    - Input: `(-5.0, 5.0, -5.0)`
    - Expected Output: `Cannot form a triangle`
    - Covered Classes: E7
    - Explanation: At least one side is negative, which cannot form a triangle.
7. Test Case 7 (Invalid Triangle - Non-Number):
    - Input: `(5.0, 'b', 5.0)`
    - Expected Output: `Cannot form a triangle`
    - Covered Classes: E8
    - Explanation: At least one side is not a valid number (string input).
8. Test Case 8 (Valid but Can't Classify):
    - Input: `(1.0, 3.0, 1.0)`
    - Expected Output: `Cannot form a triangle`
    - Covered Classes: E9
    - Explanation: The lengths do not satisfy the triangle inequality theorem since 1+1<31 + 1 < 31+1<3.

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

Test Case 1 (Valid Equilateral Triangle):

- **Input**: `(5.0, 5.0, 5.0)`
- **Expected Output**: `Equilateral`
- **Covered Classes**: E1
- **Explanation**: All three sides are equal, forming a valid equilateral triangle.

Test Case 2 (Valid Equilateral Triangle - Slide Up):

- **Input**: `(6.0, 6.0, 6.0)`
- **Expected Output**: `Equilateral`
- **Covered Classes**: E1

- **Explanation**: All three sides are equal, forming a valid equilateral triangle.

Test Case 3 (Valid Equilateral Triangle - Slide Down):

- **Input**: `(4.0, 4.0, 4.0)`
- **Expected Output**: `Equilateral`
- **Covered Classes**: E1
- **Explanation**: All three sides are equal, forming a valid equilateral triangle.

Test Case 4 (Invalid Triangle - Sum Violation):

- **Input**: `(5.0, 5.0, 11.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E5
- **Explanation**: The lengths do not satisfy the triangle inequality theorem since 5+5<115 + 5 < 115+5<11.

Test Case 5 (Invalid Triangle - Non-Positive Side):

- **Input**: `(0.0, 0.0, 0.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E6
- **Explanation**: All sides are zero, which cannot form a triangle.

Test Case 6 (Invalid Triangle - Negative Side):

- **Input**: `(-5.0, -5.0, -5.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E7
- **Explanation**: All sides are negative, which cannot form a triangle.

Test Case 7 (Invalid Triangle - Non-Number):

- **Input**: `(5.0, 'c', 5.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E8
- **Explanation**: At least one side is not a valid number (string input).

Test Case 8 (Valid but Can't Classify):

- **Input**: `(3.0, 3.0, 6.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E9
- **Explanation**: The lengths do not satisfy the triangle inequality theorem since 3+3<63 + 3 < 63+3<6.

f) For the boundary condition A2 + B2 = C2 case (right-angle triangle), identify test cases to verify the boundary.

Test Case 1 (Valid Right-Angled Triangle):

- **Input**: `(3.0, 4.0, 5.0)`
- **Expected Output**: `Right-angled`
- **Covered Classes**: E4
- **Explanation**: 32+42=9+16=25=523^2 + 4^2 = 9 + 16 = 25 = 5^232+42=9+16=25=52, valid right-angled triangle.

Test Case 2 (Valid Right-Angled Triangle - Slide Up):

- **Input**: `(5.0, 12.0, 13.0)`
- **Expected Output**: `Right-angled`
- **Covered Classes**: E4
- **Explanation**: 52+122=25+144=169=1325^2 + 12^2 = 25 + 144 = 169 = 13^252+122=25+144=169=132, valid right-angled triangle.

Test Case 3 (Valid Right-Angled Triangle - Slide Down):

- **Input**: `(6.0, 8.0, 10.0)`
- **Expected Output**: `Right-angled`
- **Covered Classes**: E4
- **Explanation**: 62+82=36+64=100=1026^2 + 8^2 = 36 + 64 = 100 = 10^262+82=36+64=100=102, valid right-angled triangle.

Test Case 4 (Invalid Triangle - Sum Violation):

- **Input**: `(2.0, 2.0, 5.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E5
- **Explanation**: The lengths do not satisfy the triangle inequality theorem since 2+2<52 + 2 < 52+2<5.

Test Case 5 (Invalid Triangle - Non-Positive Side):

- **Input**: `(0.0, 4.0, 4.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E6
- **Explanation**: At least one side is zero, which cannot form a triangle.

Test Case 6 (Invalid Triangle - Negative Side):

- **Input**: `(-3.0, -4.0, -5.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E7
- **Explanation**: All sides are negative, which cannot form a triangle.

Test Case 7 (Invalid Triangle - Non-Number):

- **Input**: `(5.0, 'h', 5.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E8
- **Explanation**: At least one side is not a valid number (string input).

Test Case 8 (Valid but Can't Classify):

- **Input**: `(1.0, 1.0, 3.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E9
- **Explanation**: The lengths do not satisfy the triangle inequality theorem since 1+1<31 + 1 < 31+1<3.

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

Test Case 1 (Invalid Triangle - Sum Violation):

- **Input**: `(1.0, 2.0, 3.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E5
- **Explanation**: The lengths do not satisfy the triangle inequality theorem since 1+2=31 + 2 = 31+2=3 (not greater).

2. **Test Case 2 (Invalid Triangle - Greater than Sum)**:
   - **Input**: `(5.0, 1.0, 3.0)`
   - **Expected Output**: `Cannot form a triangle`
   - **Covered Classes**: E5
   - **Explanation**: The lengths do not satisfy the triangle inequality theorem since 1+3<51 + 3 < 51+3<5.
3. **Test Case 3 (Invalid Triangle - Non-Positive Side)**:
   - **Input**: `(0.0, 2.0, 2.0)`
   - **Expected Output**: `Cannot form a triangle`
   - **Covered Classes**: E6

    - **Explanation**: At least one side is zero, which cannot form a triangle.
4. **Test Case 4 (Invalid Triangle - Negative Side)**:
    - **Input**: `(-1.0, 2.0, 2.0)`
    - **Expected Output**: `Cannot form a triangle`
    - **Covered Classes**: E7
    - **Explanation**: At least one side is negative, which cannot form a triangle.
5. **Test Case 5 (Invalid Triangle - Non-Number)**:
    - **Input**: `(5.0, 'a', 5.0)`
    - **Expected Output**: `Cannot form a triangle`
    - **Covered Classes**: E8
    - **Explanation**: At least one side is not a valid number (string input).
6. **Test Case 6 (Valid but Can't Classify)**:
    - **Input**: `(2.0, 2.0, 5.0)`
    - **Expected Output**: `Cannot form a triangle`
    - **Covered Classes**: E9
    - **Explanation**: The lengths do not satisfy the triangle inequality theorem since 2+2<52 + 2 < 52+2<5.
7. **Test Case 7 (All Sides Zero)**:
    - **Input**: `(0.0, 0.0, 0.0)`
    - **Expected Output**: `Cannot form a triangle`
    - **Covered Classes**: E6
    - **Explanation**: All sides are zero, which cannot form a triangle.
8. **Test Case 8 (Two Sides Zero)**:
    - **Input**: `(0.0, 0.0, 2.0)`
    - **Expected Output**: `Cannot form a triangle`
    - **Covered Classes**: E6
    - **Explanation**: At least one side is zero, which cannot form a triangle.

h) For non-positive input, identify test points.

Test Case 1 (One Negative Side):

- **Input**: `(-1.0, 2.0, 3.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E7
- **Explanation**: The first side is negative, making it impossible to form a triangle.

Test Case 2(All Negative Sides):

- **Input**: `(-1.0, -2.0, -3.0)`
- **Expected Output**: `Cannot form a triangle`

- **Covered Classes**: E7
- **Explanation**: All sides are negative, which cannot form a triangle.

Test Case 3 (Mixed Negative and Positive):

- **Input**: `(1.0, -2.0, 3.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E7
- **Explanation**: One side is negative, which cannot form a triangle.

Test Case 4 (Negative and Zero):

- **Input**: `(0.0, -2.0, 3.0)`
- **Expected Output**: `Cannot form a triangle`
- **Covered Classes**: E6, E7
- **Explanation**: One side is zero and another is negative, which cannot form a triangle.