3. Test case design

The triangle problem

Practical design of test cases
(most examples based on materials by Collard & Co.)
Lessons learned

Test cases – key premises

- We cannot test everything
- Good testing relies on a good selection of test cases

The triangle problem (simple version)

- The input to the `TriangleType` function are three numbers a, b and c that represent the lengths of the three sides of the triangle.
- Based on these inputs the function determines the type of the triangle, which can be
  - Equilateral (i.e. all three sides are equal)
  - Isosceles (two equal sides)
  - Scalene (three unequal sides)
- The function returns the result in the form of the character string, e.g. 'Equilateral' if the triangle is equilateral.

Class exercise

Suggest a set of test cases for testing the `TriangleType` function

15 minutes

Minimal test?

Test case	Input values	Expected results
1	3, 3, 3	Equilateral
2	3, 3, 2	Isosceles
3	3, 4, 5	Scalene

Automated test?

```
for i = minval to maxval by increment do
   for j = minval to maxval by increment do
      for k = minval to maxval by increment do
         enter { i, j, k }
         capture { result }
         if [ i = j = k ] and result = equilateral
           or if [ ( i = j ) and ( i not = k ) and result = isosceles ]
           or if [ ( j = k ) and ( j not = i ) and result = isosceles ]
           or if [ ( k = i ) and ( k not = j ) and result = isosceles ]
           or if [ ( i not = j ) and ( j not = k ) and ( k not = i )
           and result = scalene ]
         then
           ok
         else
           write error_log [ i, j, k, result ]
      end
   end
end
```

Automated Monte Carlo test?

```
do while we_want_to_keep_testing
        i = random ( i )
        j = random ( j )
        k = random ( k )
        enter { i, j, k }
        capture { result }
        if [ i = j = k ] and result = equilateral
           or if [ ( i = j ) and ( i not = k ) and result = isosceles ]
           or if [ ( j = k ) and ( j not = i ) and result = isosceles ]
           or if [ ( k = i ) and ( k not = j ) and result = isosceles ]
           or if [ ( i not = j ) and ( j not = k ) and ( k not = i )
           and result = scalene ]
        then
          ok
        else
           write error_log [ i, j, k, result ]
end
```

More test cases

Input values			Expected results	Actual results
-5	-5	-5	Invalid	Equilateral
1	10	1	Invalid	Isosceles
1	2	3	Invalid	Scalene
130	140	130	Isosceles	Invalid
3.329951	3.330023	3.330050	Equilateral	Scalene
4	3	3	3	Invalid	Equilateral

Lessons learned (1)

- Specification must be as precise as possible

Lessons learned (2)

- Testing must include negative cases
- Question:
  Is it useful to have more negative than positive tests?
- Answer
  It depends on
  - The required reliability
  - The "perceived hostility of the environment"

  Normally positive cases should outnumber negative

Lessons learned (3)

- Effective testing requires careful selection of appropriate test cases
- Accurate and thorough specification is essential
- It helps to have background knowledge, both of the application area to be tested and of computer programming

Homework

- Improve the Automated Monte Carlo test given in the lecture