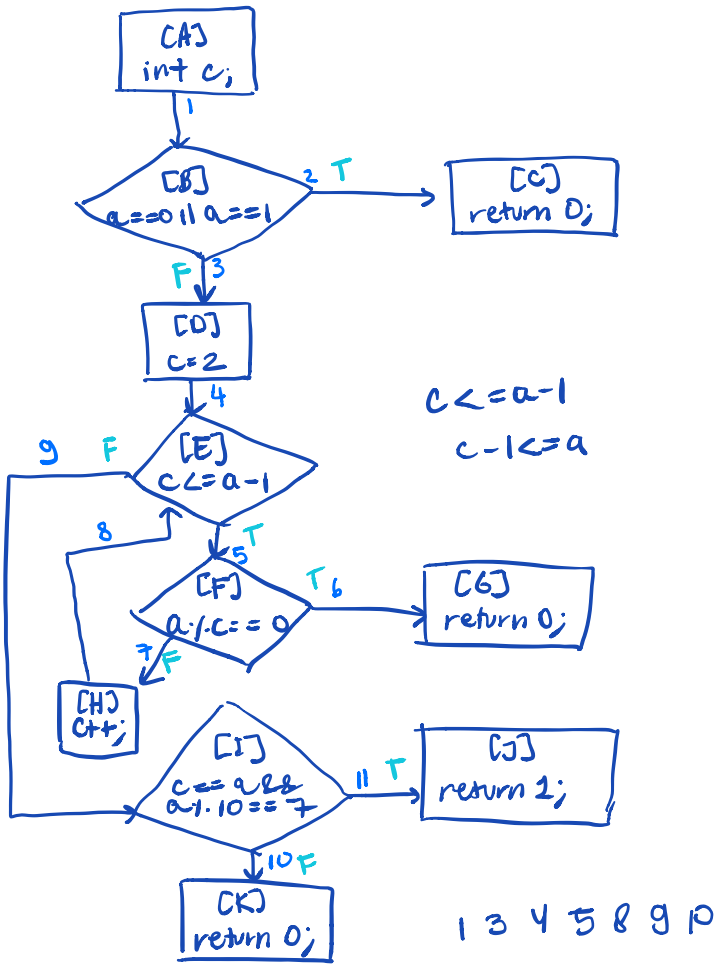
ST 2016 Past Paper

1. Method check_prime() checks if a given number is prime and if the number has 7 as its unit digit. → Returns 1 if true; 0 otherwise.

```
public int check_prime(int a)
{
   int c;
   if ( a ==0 || a ==1)
      return 0;
   for ( c = 2 ; c <= a - 1 ; c++ )
   {
      if ( a%c == 0 )
          return 0;
   }
   if ( (c == a) && (a%10 == 7) )
      return 1;

   return 0;
}
```

a. Draw CFG, label edges with numbers and blocks with letters.



b. Write tests with input and expected output. Write the seq. of edges traversed by each of the tests (disregard loops).

Test	Input	Expected output	Edges traversed
a	0	0	1, 2
b	2	0	1, 3, 4, 9, 10
c	3	0	1, 3, 4, 5, 7, 8, 9, 10
d	7	1	1, 3, 4, 5, 7, 8, 8, 11

* a = 0, b = 2, c = any prime with unit value != 7, d = any prime with unit value = 7

c. Evaluate the fraction of edges covered by tests. If any uncovered, write additional tests.

coverage = 10/11 uncovered = 7

Test	Input	Expected output	Edges traversed
e	4	0	1, 3, 4, 5, 7

any non-prime > 2

d. typo → || instead of && in if ((c == a) && (a % 10 == 7)).

This means that any prime (regardless if they're unit value is not 7) will return 1.

Test c (a = 3) will catch this. (Expected = 0, actual = 1)

e. Which coverage criterion is practical and rigorous for testing complex boolean expressions? Develop a test for ((x1 || y1) && z1).

MC/DC.

((x1 || y1) && z1) truth table

x1	y1	z1	((x1 || y1) && z1)
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

4 tests needed.

f. For the checkprime method, write tests that achieves 'all DU pairs' coverage for variable c. Write out DU pairs for c + coverage.

Variables	Definitions	Uses	DU pairs
a	A	B, E, F, I	<A, B>, <A, E>, <A, I>, <A, F>
c	A, D, H	E, F, H, I	<D, E>, <D, F>, <D, H>, <D, I>, <H, E>, <H, H>, <H, I>

Test with a = 5 covers all.

2. Function to calculate factorial (n!) for a number n, returns 0 for negative inputs and returns -1 when the result is out of range:

```
int factorial (int n)
```

a. Assuming the factorial method is the ITF - "correctly computes and returns the factorial of its parameter, n, or returns an appropriate error code" what characteristics of the input and environment for this ITF?

Input: n can be positive or negative.
Environment: int size - result can be in or out of range

b. Define partitions / value classes for the input and environment based on the characteristics

	Partition	Input
n	P1	n > 0
	P2	n 0
result	P3	in int range
	P4	out of int range

c. Generate test case specifications for diff. value classes + expected result.

Test	n	result	expected result
a	positive	in int range	correct factorial
b	positive	out of int range	-1
c	negative	in int range	0
d	negative	out of int range	0

d. concrete test cases for each test case spec + expected result. (max=10).

Test	n	integer size (bytes)	expected result
a	-3	2	0
b	0	2	1
c	3	2	6
d	-10	2	0
e	10	2	-1
f	-3	4	0
g	0	4	0
h	-10	4	0
i	10	4	10!
j	20	4	-1

3. Function that checks if a given character (arg) is a vowel. Function returns 1 if vowel, 0 otherwise.

```
1  public int check_vowel(char a)
2 {
3    if (a >= 'A' && a <= 'Z')
4       a = a + 'a' - 'A';   /* Converting to lower case  */

5    if (a == 'a' || a == 'e' || a == 'i' || a == 'o' || a == 'u')
6       return 1;

7    return 0;
8 }
```

a. Derive 6 mutations that perform expression/operand modifications.

Mutation	Line no.	Change
M1	3	a >= 'A' → a > 'A'
M2	3	a <= 'Z' → a < 'Z'
M3	4	a + 'a' - 'A' → a + 'a' + 'A'
M4	5	(a == 'a' || a == 'e' ...) → (a != 'a' && a != 'e' ...)
M5	5	a != 'a'
M6	4	a + 'a' - 'A' → a - 'a' + 'A'

b. Tests to reveal mutations. Point out equivalent mutants.
← 'A' = 65, 'Z' = 90, 'a' = 97, 'z' = 122

Mutation	Test	Expected Output	Actual output
M1	'A'	1	0
M2	'Z'	0	0 → not revealed!
M3	'A'	1	0 → gets transformed as non-alpha
M4	'Z'	0	1 → acc. consonants
M5	'a'	1	0
M6	'A'	1	0 → gets transformed to non-alpha

c. New spec: check all vowels EXCEPT 'u', add 'r' as vowel. Will above tests pass or fail?

```
1  public int check_vowel(char a)
2 {
3    if (a >= 'A' && a <= 'Z')
4       a = a + 'a' - 'A';   /* Converting to lower case  */

5    if (a == 'a' || a == 'e' || a == 'i' || a == 'o' || a == 'r')
6       return 1;

7    return 0;
8 }
```

All tests will pass because neither 'u' nor 'r' was used as a test input parameter.

d. Write tests for the new spec.

Test	Input	Expected output
a	'u'	0
b	'U'	0
c	'r'	1
d	'R'	1

e. In which testing activity are global properties, such as performance and security, that involve interactions between the system and the environment tested? Describe one method for testing such properties.

System testing → test properties of the systems.
→ can test global properties like performance, latency, reliability

* It's important that people testing this are not the ones involved with the development of the system.

Some methods: capacity testing, performance testing, stress testing.