S3:
1) CFG
CC - E - N + 2
- Predicate + 1
- nof of regions
individual paths

2) TC -> coverage -> statement, paths, condition, decision-condition, loop, multiplication, loop coverage

I. Verify if a natural nr is prime

```
function Prim(m: integer) : boolean; var i: integer, b: boolean
begin
1       b := true;
2       if m < 2 then
3               b := false
        else
                begin
4                       i := 1 (should be 2)
5                       while b and (i < m div 2 ) do
6                               if m mod i = 0 then
7                                       b := false
                                else
8                                       b := true
                                i : = i + 1 (should be)
                end
9               Prim := b
end
```

CC = E - N + 2 = 12 - 10 + 2 = 4
= Pred + 1 = 3 + 1
= nr of reg = 4

underline is what makes them unique
P1: 1 - 2 - 3 - 9
P2: 1 - 2 - 4 - 5 - 9
P3: 1 - 2 - 4 - 5 - 6 - 7 - 8' - 5 - ... - 9
P4: 1 - 2 - 4 - 5 - 6 - 8 - 8' - 5 - ... - 9

2. TC
a) Statement coverage

#TC	Statements	Input m	Output expected	Output actual
1	1, 2, 3, 9	1	false	false

2	1, 2, 4, 5, 9	2	true	true
3	1 - 2 - 4 - 5 - 6 - 7- 8' - 5 - ... - 9	4	false	false
4	1 - 2 - 4 - 5 - 6 - 8 - 8' - 5 - ... - 9	5	true	false

Remarks:
1) TC4 -> bug
2) Statemen 8' can not be covered

STEPS:
- TC 1 .. 4
- bug
- i = 2 (in statement 4) and i = i + 1 (8')
- CFG - new
- ? TC 1.. 4

?! TC5 m = 9

TC5	m = 9	F	F
bug 2 <= T1..5	m = 4 i != 4	T	F

b) Path coverage == (not always) Statement Coverag

#TC	Paths	#TC_ Statements
1	P1	1
2	P2	2
3	P3	3
4	P4	5

c) Decision coverage

#TC	S2 T	S2 F	S5 T	S5 F	S6 T	S6 F	input	output
1	X						1	F
2		X	X	X	X		4	F
3		X	X	X	X	X	9	F

d) Condition Coverage
(this is the same as the table above, this is stupid)

#TC	S2 T	S2 F	S5 C1 = b T	S5 C1 = b F	S5 C2 = i <= m div 2 T	S5 C2 = i <= m div 2 F	S6 T	S6 F	input	output expected
1	X								1	F
2		X	X	X	X		X		4	F
3‘		X	X			X			3	F

T3 here was added

TC? a) = b) = c) => 3 T + d) (3’) = 4T
3T in case 3' we make m = 7 so that we have 3 test cases instead of 4

so we have the following TC: 1, 2, 3‘
e) Loop

loop n = max

#TC	loop 0	loop 1	loop k < n	loop k = n	loop k = n + 1	Input m	Output b
1	X				this is not possible	1	F
2		X		X		4 (n is 1)	F
deleted 3				X		7 (n is 2)	T
3			X			9	F

d) + e) => 3 Td + 1 Te => 3‘’
II.

```
procedure CeaMaiLungaSecventa (nr: Integer, valE: sirnat, var pozF, lungf: Integer)
begin
        pozF = 0; lungF = 0; pozI := 0; lungI := 0; i :=0;
        while (i <= nrE) do
        begin
                if EstePrim ( valE(i) - valE(i + 1)) then
                begin


end
```

graf in caiet
CN = no of regions = 5 = Pred (while, if + 1

Individual path = 5

P1: 1 - 2 - 9
P2: 1 - 2 - 3 - 4 - 5 - 8 - 2 ... 9
P3: 1 - 2 - 3 - 4 - 6 - 7 - 8 ... 9
P4: 1 - 2 - 3 - 4 - 6 - 8 ... 9
P5: 1 - 2 - 3 - 8 ... - 9

Fixed bugs:
i := 1
while i < nrE
abs(valE(i) - valE(i + 1))
the second if should be in the first
must increase lungI somewhere

#TC	Path	input nrE	input valE	output pozF	output lungF	actual	actual
1	P1	0	[]			bug	bug
2	P2	2	[5, 8]	1	2	0 (bug)	0 (bug)
3	P3	3	[5, 8, 5]	1	3	1	2
4	P4	5	[5, 8, 9, 5, 8]	1	2	1 (bug)	4 (bug)