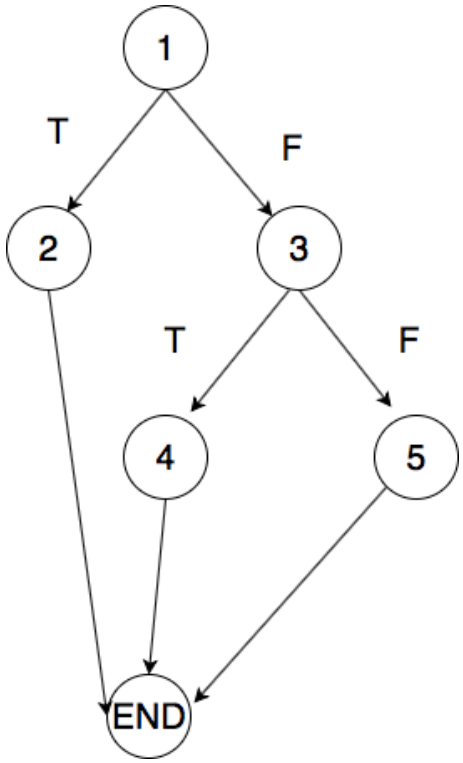
Coursework 2

Name: Chong Yang Number: 18089022

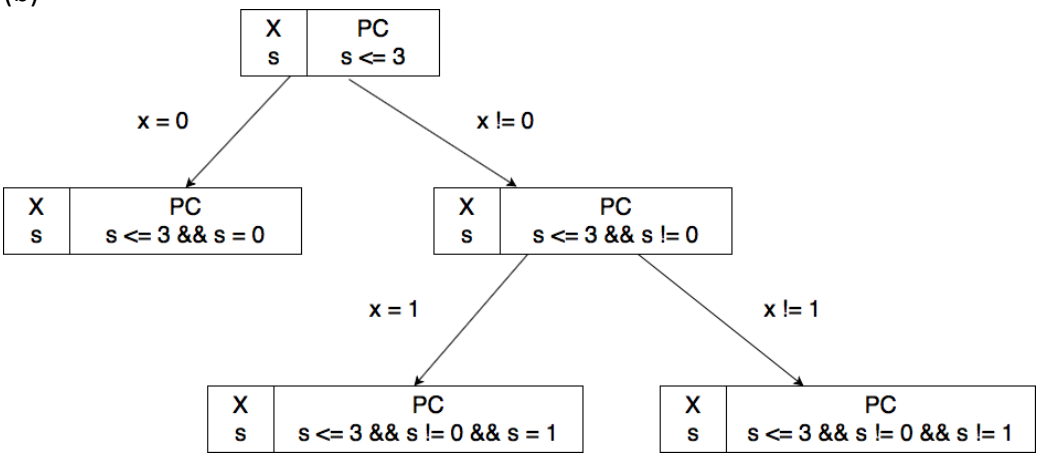
1.

(a)

```
    int factorial( nat x ) {
1       if ( x == 0 )
2           return 1;
3       if ( x == 1 )
4           return 1;
        else
5       return x*factorial(x – 1);
```



(b)



2.
Assertions could be added to check whether a statement holds based on the variable values in a specific path in symbolic execution. The program will terminate and raise the error if the assertion is false.

3.
Symbolic execution is to use symbolic values, instead of actual data, as input values, and to represent the values of program variables to generate testing. The input structures and constraints can be used for test input generation.

4.
a) Solver unknowns: SMT solver returns an unknown result.

b) External functions: External functions are required. External functions are functions that are external to a symbolic execution engine, because that engine cannot symbolically execute them.

c) Path explosion: The program only explores the interesting paths. Loop invariant could be handled.

d) Source code requirement.

e) Intractable constraints.

5.
No.
Verification is executed during development process and testing after development. Verification is the process of resolving bugs and issues during development while testing is the process of checking "is developed functionality as per requirement or not." Therefore verification should be done before testing. Moreover, verification should be done by developer and software testing should be done by tester.

6.
(a)
Outside the loop, a is bound to $s_1 = 1$ and b is bound to $s_2 = 5$, so then the body of the first if executes once while the second if never executes.

$$PC = s_1 = 1 \ \wedge \ s_2 = 5$$
$$\wedge (s_1 \geq 0 \ \wedge \ s_2 \leq 7)$$
$$\wedge (s_1 - 1 \geq 0 \ \wedge \ s_2 + 1 \leq 7)$$
$$\wedge (s_1 - 2 < 0 \ \wedge \ s_2 + 2 \leq 7)$$

(b)

$x = (s_1 - 2) + (s_2 + 2) = 6$

7.

(| T |)
a = x – k;
(| a = x – k |) Assignment
if (a + k == k) { (a + k = k and a = 0)
  (| 0 = x – k|) Implied
  y = 0;
  (|y = x - k|) Assignment
}
else { (a + k != k)
  (| a = x – k |) Implied
  y = a;
  (|y = x - k|) Assignment
}
(|y = x - k|) If-statement

8.

(a)

False.

Counterexample:

P = x > 0
R = x < 5
C = x := x + 1
Q = x > 0 ∧ x < 6

Then, (P∧R) C (Q) is (x > 0 ∧ x < 5) x := x + 1 (x > 0 ∧ x < 6), which is valid.
However, (x > 0) x := x + 1 (x > 0 ∧ x < 6) is not valid.

(b)

True.

If <P> C <Q Λ R> is valid, it equally means if P is true and C is executed, then Q Λ R is true. Q Λ R is true means Q and R are true separately. Consequently if P is true and C is executed, then Q is true. Therefore, <P> C <Q> is valid.

9.

(a)

If the program does not terminate, the Hoare triple is:

1) partial correctness.
2) not total correctness except the precondition is false.

(b)
(x = 1) while (true) do {skip;} (x = 2)

(c)
(x = 1) x := x + 1 (x > 0)

10.
(a)
y < 10

(b)
$5y^2 - zy + z > 0$

11.
(| T |)
i = 0;
res = 3;
$(|res = 2^1 + 1 = 2^{2^0} + 1 = 2^{2^i} + 1|)$ Assignment
while (i < n) { (| i != n |)
  $(|res = 2^{2^i} + 1|)$ Implied
  res = (res – 2) * res + 2;
  $(|\ res = (res - 1)^2 + 1 = (2^{2^i} + 1 - 1)^2 + 1 = (2^{2^i})^2 + 1 = 2^{2^{i+1}} + 1\ |)$ Implied
  $(|\ res = 2^{2^{i+1}} + 1|)$ Assignment
  i = i + 1;
  $(|res = 2^{2^{(i-1)+1}} + 1 = 2^{2^i} + 1|)$ Assignment
}
$(|res = 2^{2^i} + 1$ and i = n$|)$ Partial-While
$(|res = 2^{2^n} + 1\ |)$ Implied