# CS 536 – Science of Programming

## Assignment – 5

1)

a) **Full proof outline under partial correctness :-**

```
{n>0} k:=n-1; {n>0 ∧ k=n-1} x:=n; {n>0 ∧ k=n-1
                                    ∧ x=n}

{inv p ≡ 1 ≤ k ≤ n ∧ x = n! ÷ k!}
while k > 1 do
{p ∧ k > 1}
    {p[x * k/x]{k-1/k]} k:=k-1; {p{x * k/x]}
                                    x:=x * k {p}
od
{p ∧ k ≤ 1} {x = n!}
```

b) **Minimal proof outline under partial correctness :-**

```
{n>0} k:=n-1; x:=n;
{inv p ≡ 1 ≤ k ≤ n ∧ x = n! ÷ k!}
while k > 1 do
    k:=k-1; x:=x * k
od
{x = n!}
```

2) Full proof outline under partial correctness :-

```
{n ≥ 0}
{p {0|k] [0/s]}
k := 0;
{p [0/s]}
s := 0;
{inv p ≡ 0 ≤ k ≤ n ∧ s = sum(0,k)}
while k < n do
 {p ∧ k < n}
    s := s+k+1;
    {0 ≤ k ≤ n ∧ s0 = sum(0,k) ∧ s =
                                    s0 + k+1}

    k := k+1;
    {0 ≤ k0 ≤ n ∧ s0 = sum(0,k0) ∧ s =
                                    s0 + k0+1
                               ∧ k := k0+1}

{p}
od
{p ∧ k ≥ n}
{s = sum(0,n)}
```

3) <u>Full proof outline under partial correctness</u> :-

```
$\{y \geq 1\}$
$x := 0; \{y \geq 1 \wedge x = 0\}$
$r := 1; \{y \geq 1 \wedge x = 0 \wedge r = 1\}$
$\{ \text{inv } p \equiv 1 \leq r = 2^{x} \leq y \}$
while $2 * r \leq y$ do
  $\{p \wedge 2 * r \leq y\}$
    $r := 2 * r; \{1 \leq r_0 = 2^{x} \leq y \wedge 2 * r_0 \leq y \wedge r = 2 * r_0\}$
    $x := x + 1; \{1 \leq r_0 = 2^{x_0} \leq y \wedge 2 * r_2 \leq y \wedge r = 2 * r_0 \wedge x = x_0 + 1\}$

  $\{1 \leq r = 2^{x} \leq y\}$
  od
$\{p \wedge 2 * r > y\}$
$\{r = 2^{x} \leq y \leq 2^{(x+1)}\}$
```

<u>Exaplanation of each logical implication used in the proof outline</u> :-

* With the condition $y \geq 1 \wedge x = 0 \wedge r = 1$

the loop invariant p is satisfied which allows us to begin the loop.

$* \; P_1 \Rightarrow P$: since initially $1 \le x_0 = 2^{x_0}$ and after the first iteration, $x = 2 * x_0 \wedge x = x_0 + 1$, it follows that $1 \le x = 2^x$.
Given that $2 * x_0 \le y$ and $x = 2 * x_0$.
we also have that $x \le y$.

$* \; P \wedge (2 * x > y) \Rightarrow (x = 2^x \le y < 2^{x+1})$,
since $P \equiv 1 \le x = 2^x \le y$, so $x = 2^x \le y$.
since $2 * x > y$, so $y < 2^{x+1}$

4) Finding a bound expression for the while loop :-

The while loop after observing it we can say that y-x can be one of the bound expression.

Full proof outline under total correctness for the partial proof with Backward Assignment :-

$\{y \ge 1\}$
$\{1 \le 1 = 2^0 \le y\}$ $x := 0;$
$\{1 \le 1 = 2^x \le y\}$ $x := 1;$

```
{inv $p \equiv 1 \leq x = 2^x \leq y$ } { bd $y - x$ }
while $2 * x \leq y$ do
    { $p \wedge 2 * x \leq y \wedge y - x = t_0$ }
        { $1 \leq 2 * x = 2^{x+1} \leq y \wedge y - 2 * x < t_0$ }
        $x := 2 * x$;
        { $1 \leq x = 2^{x+1} \leq y \wedge y - x < t_0$ }
        $x := x + 1$;
{ $1 \leq x = 2^x \leq y \wedge y - x < t_0$ }
od
{ $p \wedge 2 * x > y$ }
{ $x = 2^x \leq y \leq 2^{(x+1)}$ }
```

5) Given provable triple,

{P} if sqrt(x) > y then x := b[x − y] else y := b[y − x] fi {x = y}

Full proof outline for the above given provable triple:-

```
{P}
if sqrt(x) > y then
```

$\{b[x-y] = y \wedge 0 \leq (x-y) < size(b)\}\ x := b[x-y]$

$\{x = y\}$

else

$\{x = b[y-x] \wedge 0 \leq (y-x) < size(b)\}\ y := b[y-x$

$\{x=y\}$

fi

$\{x=y\}$

where $P \equiv \{x \geq 0 \wedge (sqrt(x) > y \rightarrow b[x-y] = y)$
$\wedge (sqrt(x) \leq y \rightarrow x = b[y-x])\}$

6) Given provable triple,

$\{sqrt(x) \leq y\}\ x := x * y;\ x := 1 \div x \{q\}$

Full proof outline for the above given provable triple

using Forward assignment,

$\{sqrt(x) \leq y \wedge x \geq 0\}\ x := x * y$

$\{sqrt(x_0) \leq y \wedge x_0 \geq 0 \wedge x = x_0 * y\}$

$x := 1 \div x$

$\{ \mathrm{sqrt}(x_0) \leq y \wedge x_0 \geq 0 \wedge x_1 = x_0 * y \wedge x =$
$1 \div x_1 \wedge x_1 \neq 0 \}$

$q \equiv \{ \mathrm{sqrt}(x_0) \leq y \wedge x_0 \geq 0 \wedge x_1 = x_0 * y$
$\wedge x = 1 \div x_1 \wedge x_1 \neq 0 \}$

7)

a) True, Because if $\sigma$ satisfies the invariant p and the loop terminates correctly without any divergence or errors, then no undefined behaviour occurs during the execution of loop (not diverge).

b) False, The bound function is typically used to ensure termination and is non-negative. It decreases with each iteration but cannot become negative after the last iteration.

c) True, Because if we start with a state where the invariant holds, and the bound function equals some value to, after executing one iteration of the loop body, the value of the bound function should decrease. This is a standard requirement for termination.

d) False, Because the statement implies that if the invariant holds and the bound function is positive, then the loop condition must still hold. However, this is incorrect because even if it is $t > 0$, it does not necessarily mean that the loop condition will still be true.

e) True, The bound function cannot be negative during execution. Hence, if it were somehow negative, this would contradict our assumption that the invariant holds

8)

a) It cannot be a bound expression.
Because a bound function must always be non-negative and decrease with each iteration

There is no evidence to show that $x-k+n \geq 0$ thorought the loop. So, it can't be a bound expression.

b) It can be a bound expression.

The expression $n-k$ decreases with each iteration because k is incremented by 1 in each loop iteration. Also, since the loop stops when $k > n$, this expression will reach zero which makes it a valid bound function.

c) It can be a bound expression.

The loop invariant implies that $n-k+c \geq 0$ and since k increases after each iteration the expression $n-k+c$ decreases over time. Therefore, this can be a valid bound expression.

d) It cannot be a bound expression. The value of K increases after each iteration, so the expression K-C will also increase after each iteration. A bound function must decrease with each iteration. So this can't be a valid bound function.

e) $2^n \cdot 2^{C-K}$

It can be a bound expression. Since, both n and C are constants and only K increases with each iteration, this expression decreases as K increases. It is non-negative and decreases with each iteration, making it a valid bound function.

9) Five possible candidates for the loop invariant P and their corresponding loop condition B :-

Here u is a new variable which we use to replace constants in the predicate.

$q \equiv y \geq 0 \land x = 2*y \leq n < 3*(y+1)$

1) $P_1 \equiv y \geq u \wedge x = 2 * y \leq n < 3 * (y+1)$ and

$B_1 \equiv u \neq 0$

2) $P_2 \equiv y \geq 0 \wedge x = u * y \leq n < 3 * (y+1)$ and

$B_2 \equiv u \neq 2$

3) $P_3 \equiv y \geq 0 \wedge x = 2 * y \leq n < u * (y+1)$ and

$B_3 \equiv u \neq 3$

4) $P_4 \equiv y \geq 0 \wedge x = 2 * y \leq n < 3 * (y+u)$

and $B_4 \equiv u \neq 1$

5) $P_5 \equiv y \geq 0 \wedge x = u * y \leq n < 3 * (y+1)$

and $B_5 \equiv u \neq 2$

10) Four possible candidates for the loop invariant P and thier corresponding loop condition B :-

1) $P_1 \equiv (z = 2^y) \wedge (2^y \leq x) \wedge (x < 2^{y+1})$ and

$B_1 \equiv y < 0$

2) $P_2 \equiv (y \geq 0) \wedge (2^y \leq x) \wedge (x < 2^{y+1})$ and

$B_2 \equiv z \neq 2^y$

3) $P_3 \equiv (y \geq 0) \wedge (z = 2^y) \wedge (x < 2^{y+1})$ and

$B_3 \equiv 2^y > x$

4) $P_4 \equiv (y \geq 0) \wedge (z = 2^y) \wedge (2^y \leq x)$ and

$B_4 \equiv x \geq 2^{y+1}$.