Software Quality Assurance (WS20/21)
Problem Set 6

Problem 1: Theory

a) Briefly explain the difference between partial and total correctness.

-

Partial correctness: if the algorithm returns a result, it is correct.
Total correctness: a result exists and algorithm terminates (one is always delivered).

b) Does the Hoare calculus prove partial or total correctness?

-

The Hoare calculus is used to show partial correctness.

c) Give suitable post conditions for the following statements or assertions, taking into account the preconditions.

- $x=\sqrt{y}$; precondition: $y \geq 0 \wedge y \in R$

Post condition: $x \geq 0 \wedge x \in R \wedge x^2 = y$

- int a[] = sort(b[]);
  precondition: sizeof(a[]) = sizeof(b[]) ∧ ∀i, n∈N " with " 0≤i≤n " holds " b[i]∈Z

Post condition:
∀i, n ∈ N with " 0 ≤ i ≤ n " holds a[i], b[i] ∈ Z ^
∀i, n ∈ N with " 0 ≤ i ≤ n " holds a[i + 1] ≥ a [i]

Problem 2: Symbolic Execution Testing

Fill the tables by performing a symbolic execution test for the following program, then conduct the terms which represent the behavior of the program

-

```
void test_me (int x, int y){

      int z = 2 *y;
      if (z==x) {
            x= 30+y;
      } else {
            x= 30-y;
      }

}
```

X	Y	Z	PC	Variable name
X0	Y0	2*Y0	-	Symbolic Value
X0	Y0	2*Y0	2*Y0 = X0	
X0 = 30 + YO	Y0	2*Y0	2*Y0 = X0	

X	Y	Z	PC	Variable name
X0	Y0	2*Y0	-	Symbolic Value
X0	Y0	2*Y0	2*Y0 <> X0	
X0 = 30 - YO	Y0	2*Y0	2*Y0 <> X0	

((X=30+y) ^ (2*y=x)) v ((x=30-y) ^ (2*y<>x))
Since we have then explored all feasible paths and not reached an error, we can conclude that the program will not raise an error in any execution.

Problem 3: Hoare Calculus

Verify the following programs using the Hoare calculus. For each step, state the axiom you used

a)

```
void swap(int x, int y){

      x = x + y;
      y = x - y;
      x = x - y;
}
```

-

Step 1: set up the pre- and post-condition

x = A ^ y = B (PRE)

x = x + y

y = x - y

x = x - y

x = B ^ y = A (POST)

Step 2: application of A0 (assignment axiom)

x = A ^ y = B (PRE)

x = x + y

y = x – y

x - y = B ^ y = A

x = x - y

x = B ^ y = A (POST)

Step 3: application of A0 (assignment axiom)

x = A ^ y = B (PRE)

x = x + y

x - (x - y) = B ^ x - y = A

y = x – y

x - y = B ^ y = A

x = x - y

x = B ^ y = A (POST)

Step 4: application of A0 (assignment axiom)

x = A ^ y = B (PRE)

x + y - (x + y - y) = B ^ x + y - y = A

x = x + y

x - (x - y) = B ^ x - y = A

y = x – y

x - y = B ^ y = A

x = x - y

x = B ^ y = A (POST)

Step 5: Check if precondition and the last generated clause match

x + y - (x + y - y) = B ^ x + y - y = A $\Longleftrightarrow$

x + y - x = B ^ x = A $\Longleftrightarrow$

y = B ^ x = A

b) Verify the following programs using the Hoare calculus. For each step, state the axiom you used

-

```
int quad(int n){
      int i, k, y;
      i = 0;
      k = -1;
      y = 0;
      while (i < n){
            i = i + 1;
            k = k + 2;
            y = y + k;
      }
      return y;
}
```

Step 1: Set up pre-condition and post-condition

n ≥ 0 (pre)
i = 0;
k = -1;
y = 0;
while (i < n) {
i = i + 1;
k = k + 2;
y = y + k;
}
y = n^2 (post)

Step 2: Application of A4(assignment axiom)

n ≥ 0 (pre)
i = 0;
k = -1;
y = 0;
{invariant P}
while (i < n) {
{P ^ loop condition B}
i = i + 1;
k = k + 2;
y = y + k;
{P}
}
{P ^ ¬B}
y = n^2 (post)

$$\frac{P \wedge B \{S\} P}{P \{\text{WHILE B DO S END}\} P \wedge \neg B}$$

with $B = \{i < n\}$ and $P = \{ i \leq n \wedge k = 2i - 1 \wedge y = i^2 \}$

Step 3: Application of A0(assignment axiom)

n ≥ 0 (pre)
i = 0;
k = -1;
y = 0;
$i \leq n \wedge k = 2i - 1 \wedge y = i^2$
while (i < n) {
$i \leq n \wedge k = 2i - 1 \wedge y = i^2 \wedge i < n$
i = i + 1;
k = k + 2;
$i \leq n \wedge k = 2i - 1 \wedge y + k = i^2$
y = y + k;
$i \leq n \wedge k = 2i - 1 \wedge y = i^2$
}
$i \leq n \wedge k = 2i - 1 \wedge y = i^2 \wedge \neg (i < n)$
y = n^2 (post)

To find the invariant, it is helpful to create a table with the variables and their assignment for several loop iterations

Variable	*i*	*k*	*y*	*n*
Loop entry	*0*	*-1*	*0*	*5*
1. Iteration	*1*	*1*	*1*	*5*
2. Iteration	*2*	*3*	*4*	*5*
3. Iteration	*3*	*5*	*9*	*5*
4. Iteration	*4*	*7*	*16*	*5*
5. Iteration	*5*	*9*	*25*	*5*

$\{ i \leq n \wedge k = 2i - 1 \wedge y = i^2 \}$

Step 4: Application of A0(assignment axiom)

$n \geq 0$ *(pre)*
i = 0;
k = -1;
y = 0;
$i \leq n \wedge k = 2i - 1 \wedge y = i^2$
while (i < n) {
$i \leq n \wedge k = 2i - 1 \wedge y = i^2 \wedge i < n$
i = i + 1;
$i \leq n \wedge k + 2 = 2i - 1 \wedge y + k + 2 = i^2$
k = k + 2;
$i \leq n \wedge k = 2i - 1 \wedge y + k = i^2$
y = y + k;
$i \leq n \wedge k = 2i - 1 \wedge y = i^2$
}
$i \leq n \wedge k = 2i - 1 \wedge y = i^2 \wedge \neg (i < n)$
$y = n^2$ *(post)*

Step 5: Application of A0(assignment axiom)

$n \geq 0$ *(pre)*
i = 0;
k = -1;
y = 0;
$i \leq n \wedge k = 2i - 1 \wedge y = i^2$
while (i < n) {
$i \leq n \wedge k = 2i - 1 \wedge y = i^2 \wedge i < n$
$i + 1 \leq n \wedge k + 2 = 2(i+1) - 1 \wedge y + k + 2 = (i+1)^2$
i = i + 1;
$i \leq n \wedge k + 2 = 2i - 1 \wedge y + k + 2 = i^2$
k = k + 2;
$i \leq n \wedge k = 2i - 1 \wedge y + k = i^2$
y = y + k;
$i \leq n \wedge k = 2i - 1 \wedge y = i^2$
}
$i \leq n \wedge k = 2i - 1 \wedge y = i^2 \wedge \neg (i < n)$
$y = n^2$ *(post)*

Step 6: Check if the precondition and the last generated clause match
$(i + 1 \leq n) \wedge (k + 2 = 2(i+1) - 1) \wedge (y + k + 2 = (i+1)^2) \Longleftrightarrow (i \leq n) \wedge (k = 2i - 1) \wedge (y = i^2) \wedge (i<n)$
the individual partial terms of the left and right sides connected by conjunction are now tested separately
$k + 2 = 2 (i+1) - 1 \Longleftrightarrow k+2 = 2i + 2 - 1 \Longleftrightarrow k = 2i - 1$

$y + k + 2 = (i+1)^2 \Longleftrightarrow y + k + 2 = i^2 + 2i + 1$
$\Longleftrightarrow y + k = i^2 + 2i - 1 \Longleftrightarrow y = i^2 \wedge k = 2i - 1$

$i \leq n \wedge i < n \Longleftrightarrow i < n \Longleftrightarrow i + 1 \leq n$

Step 7: Application of A0(assignment axiom)

$n \geq 0$ *(pre)*
i = 0;
k = -1;
$i \leq n \wedge k = 2i - 1 \wedge 0 = i^2$
y = 0;
$i \leq n \wedge k = 2i - 1 \wedge y = i^2$
while (i < n) {
$i \leq n \wedge k = 2i - 1 \wedge y = i^2 \wedge i < n$
i = i + 1;
$i \leq n \wedge k + 2 = 2i - 1 \wedge y + k + 2 = i^2$
k = k + 2;
$i \leq n \wedge k = 2i - 1 \wedge y + k = i^2$
y = y + k;
$i \leq n \wedge k = 2i - 1 \wedge y = i^2$
}
$i \leq n \wedge k = 2i - 1 \wedge y = i^2 \wedge \neg (i < n)$
$y = n^2$ *(post)*

Step 8: Application of A0(assignment axiom)

$n \geq 0$ *(pre)*
i = 0;
$i \leq n \wedge -1 = 2i - 1 \wedge 0 = i^2$
k = -1;
$i \leq n \wedge k = 2i - 1 \wedge 0 = i^2$
y = 0;
$i \leq n \wedge k = 2i - 1 \wedge y = i^2$
while (i < n) {
$i \leq n \wedge k = 2i - 1 \wedge y = i^2 \wedge i < n$
i = i + 1;
$i \leq n \wedge k + 2 = 2i - 1 \wedge y + k + 2 = i^2$
k = k + 2;
$i \leq n \wedge k = 2i - 1 \wedge y + k = i^2$
y = y + k;
$i \leq n \wedge k = 2i - 1 \wedge y = i^2$
}
$i \leq n \wedge k = 2i - 1 \wedge y = i^2 \wedge \neg (i < n)$
$y = n^2$ *(post)*

Step 9: Application of A0(assignment axiom)

$n \geq 0$ *(pre)*

$0 \leq n \land -1 = 2*0 - 1 \land 0 = 0^2$

i = 0;

$i \leq n \land -1 = 2i - 1 \land 0 = i^2$

k = -1;

$i \leq n \land k = 2i - 1 \land 0 = i^2$

y = 0;

$i \leq n \land k = 2i - 1 \land y = i^2$

while (i < n) {

$i \leq n \land k = 2i - 1 \land y = i^2 \land i < n$

i = i + 1;

$i \leq n \land k + 2 = 2i - 1 \land y + k + 2 = i^2$

k = k + 2;

$i \leq n \land k = 2i - 1 \land y + k = i^2$

y = y + k;

$i \leq n \land k = 2i - 1 \land y = i^2$

}

$i \leq n \land k = 2i - 1 \land y = i^2 \land \neg (i < n)$

$y = n^2$ *(post)*

Problem 4: Termination Proof

Given are two jars, a game jar filled with white and black beans and a storage jar filled with a (theoretically) unlimited amount of black beans. A player changes the contents of the jars by a sequence of moves. Each move proceeds as follows:

1. Blindly draw two beans from the game jar.
2. If they are the same colour, throw both of them away and put a bean from the storage jar into the game jar.
3. Otherwise, throw the black bean away and put the white one back into the game jar.

The procedure ends when there is only one bean left in the game jar.

a) Prove the termination of the game.

-

With each move, first two beans are removed from the game jar and then one is put back. The number of beans in the game jar therefore decrease by one per turn. If there are n beans in the game jar at the beginning, the game ends after n-1 moves.

b) If the game ends, what colour is the last bean in the game box? Give reasons for your answer.

-

No useful statement can be made about the number of black beans in the course of a game, because the procedure is non-deterministic. Thus, intermediate states depend on chance, given the same starting conditions, since the game is drawn blindly. For the number of while beans, however the following applies: Either it decreases by 2 in one move if two while beans were drawn, or it remains constant in all the other cases. So the parity of the white beans is not affected by a move. Consequently, the last bean is white if the number of white beans was initially odd or black if the number of white beans was initially even.