CSC236H – Proof of Algorithm Correctness

1. Give a proof of correctness for the program below with respect to the given specification.

   - **Precondition:** A is a non-empty list of numbers.
   - **Postcondition:** Returns the average of the numbers in A.

```
def avg(A):
1.   s = 0
2.   i = 0
3.   while i < len(A):
4.         s += A[i]
5.         i += 1
6.   return s / len(A)
```

Note: A variable v subscripted with a natural number k denotes the value of the variable v at the end of the k-th iteration of the loop (if such an iteration exists).

Step 1 (Formulate a loop invariant):
Let $LI(k)$ denote the assertion that if the loop is executed at least k times, then
(a) $s_k = \sum_{j=0}^{i_k-1} A[j]$.
(b) $0 \leq i_k \leq len(A)$.

Step 2 (Prove the LI):
Basis: On entering the loop, $s_0 = 0$, and $i_0 = 0$. [Lines 1 & 2]
Therefore $0 \leq i_0 \leq len(A)$.
Also, $\sum_{j=0}^{-1} A[j] = 0$ (By convention, the empty sum is evaluated to 0).
So $s_0 = \sum_{j=0}^{-1} A[j]$.

Induction Step: Let k be an arbitrary natural number, and assume that $LI(k)$ holds.
That is, if the loop is executed at least k times, then (i) $s_k = \sum_{j=0}^{i_k-1} A[j]$, (ii) $0 \leq i_k \leq len(A)$. **[IH]**
WTP: $LI(k+1)$ holds.

We have

$$
\begin{aligned}
s_{k+1} &= s_k + A[i_k] && \text{[Line 4]} \\
&= \sum_{j=0}^{i_k - 1} A[j] + A[i_k] && \text{[IH(i)]} \\
&= \sum_{j=0}^{i_k} A[j] \\
&= \sum_{j=0}^{i_{k+1}-1} A[j] && \text{[Since, by Line 5, } i_{k+1} = i_k + 1\text{]}
\end{aligned}
$$

as wanted for part (a) in $LI(k+1)$.

Note that since there is an iteration, the condition on Line 3 must hold before the iteration. Thus $i_k < len(A)$, or equivalently, $i_k + 1 \le len(A)$. So we have

$$
\begin{aligned}
0 &\le i_k && [IH(ii)] \\
&< i_{k+1} && \text{[Since by Line 5, } i_{k+1} = i_k + 1\text{]} \\
&\le len(A)
\end{aligned}
$$

as wanted for part (b) in $LI(k+1)$.

Step 3 (Prove partial correctness):
Suppose the precondition holds and the program terminates. Since the program terminates, the loop is executed a finite number of times, say t. Consider the values of s_t, i_t on exit.
By part (b) in LI, $i_t \le len(A)$.
By exit condition, $i_t \ge len(A)$.
Hence $i_t = len(A)$. By (a) in LI, $s_t = \sum_{j=0}^{len(A)-1} A[j]$, which is the sum of all elements in A.
Therefore Line 6 returns the sum of elements in A divided by the number of elements in A, which is the average numbers in A.

Step 4 (Find an appropriate loop measure):
Let $m_k = len(A) - i_k$.

Step 5 (Prove that the loop measure is a natural number on entering the loop and after every iteration, and decreases with every iteration):

By part (b) in LI, $i_k \le len(A)$. So $m_k = len(A) - i_k \ge 0$. Thus m_k is always a natural number.

$$
\begin{aligned}
m_{k+1} &= len(A) - i_{k+1} && \text{[definition of } m_{k+1}\text{]} \\
&= len(A) - (i_k + 1) && \text{[Line 5]} \\
&= len(A) - i_k - 1 \\
&= m_k - 1 && \text{[definition of } m_k\text{]} \\
&< m_k.
\end{aligned}
$$

Thus m is always decreasing. Therefore the values of m form a decreasing sequence of natural numbers.

2. Give a proof of correctness for the program below with respect to the given specification.

    - **Precondition:** $n \in \mathbb{N}$
    - **Postcondition:** Returns n^2.

```
def Sq(n):
1.   s = 0; d = 1; i = 0
2.   while i < n:
3.       s = s + d
4.       d = d + 2
5.       i = i + 1
6.   return s
```

Note: A variable v subscripted with a natural number k denotes the value of the variable v at the end of the k-th iteration of the loop (if such an iteration exists).

Step 1 (Formulate a loop invariant):
Let $LI(k)$ denote the assertion that if the loop is executed at least k times, then
(a) $s_k = i_k^2$.
(b) $d_k = 2i_k + 1$.
(c) $0 \leq i_k \leq n$.

Step 2 (Prove the LI):
Basis: On entering the loop, $s_0 = 0, d_0 = 1$, and $i_0 = 0$. [Line 1]
Therefore $s_0 = i_0^2, d_0 = 2i_0 + 1$, and $0 \leq i_0 \leq n$.

Induction Step: Let k be an arbitrary natural number, and assume that $LI(k)$ holds.
That is, if the loop is executed at least k times, then (i) $s_k = i_k^2$, (ii) $d_k = 2i_k + 1$, (iii) $0 \leq i_k \leq n$. **[IH]**
WTP: $LI(k + 1)$ holds.

We have

$$\begin{aligned}
s_{k+1} &= s_k + d_k && \text{[Line 3]} \\
&= i_k^2 + 2i_k + 1 && \text{[IH(i) and (ii)]} \\
&= (i_k + 1)^2 \\
&= i_{k+1}^2 && \text{[Line 5]}
\end{aligned}$$

as wanted for part (a) in $LI(k + 1)$.

Also,

$$\begin{aligned}
d_{k+1} &= d_k + 2 && \text{[Line 4]} \\
&= 2i_k + 1 + 2 && \text{[IH(ii)]} \\
&= 2(i_k + 1) + 1 \\
&= 2i_{k+1} + 1 && \text{[Line 5]}
\end{aligned}$$

as wanted for part (b) in $LI(k + 1)$.

Note that since there is an iteration, the condition on Line 2 must hold before the iteration. Thus $i_k < n$, or equivalently, $i_k + 1 \le n$. So we have

$$\begin{aligned} 0 &\le i_k && [IH(iii)] \\ &< i_{k+1} && [\text{Since by Line 5, } i_{k+1} = i_k + 1] \\ &\le n \end{aligned}$$

as wanted for part (c) in $LI(k+1)$.

Step 3 (Prove partial correctness):
Suppose the precondition holds and the program terminates. Since the program terminates, the loop is executed a finite number of times, say t. Consider the values of s_t, d_t, i_t on exit.
By part (c) in LI, $i_t \le n$.
By exit condition, $i_t \ge n$.
Hence $i_t = n$. By (a) in LI, $s_t = i_t^2 = n^2$.
Therefore, by line 6, $s_t = n^2$ is returned.

Step 4 (Find an appropriate loop measure):
Let $m_k = n - i_k$.

Step 5 (Prove that the loop measure is a natural number on entering the loop and after every iteration, and decreases with every iteration):

By part (c) in LI, $i_k \le n$. So $m_k = n - i_k \ge 0$. Thus m_k is always a natural number.

$$\begin{aligned} m_{k+1} &= n - i_{k+1} && [\text{definition of } m_{k+1}] \\ &= n - (i_k + 1) && [\text{Line 5}] \\ &= n - i_k - 1 \\ &= m_k - 1 && [\text{definition of } m_k] \\ &< m_k. \end{aligned}$$

Thus m is always decreasing. Therefore the values of m form a decreasing sequence of natural numbers.