Homework 5

Question 1

Consider the following algorithm, which takes as input $n \in \mathbb{N}$:

Initialize variables $j = n$ and $s = 0$.

While $j > 0$, do the following:

Increment s by j

Decrement j by 1

Output the value of s.

Your task below is to prove (across several parts) that the above algorithm computes the sum of the first positive n integers. Use the following:

$$\textbf{Pre-condition: } n \in \mathbb{N}^+ \ \wedge \ j = n \ \wedge \ s = 0$$

$$\textbf{Post-condition: } s = \sum_{i=1}^{n} i$$

$$\textbf{Loop invariant } I : s = \sum_{i=j+1}^{n} i \ \wedge \ j \in \mathbb{N}$$

Prove that the pre-condition implies I.

If j is equal to any positive integer and $s = 0$, this implies that j will also be in $\mathbb{N}$. Additionally, if $n \in \mathbb{N}^+$, it implies that the loop will result in a real number.

Prove that if I and the loop condition both hold right before an iteration of the loop, then I holds right after the iteration.

Assuming that I and the loop condition hold before an iteration, we can start with the random values: $j = n$ and $s = 0$. After the iteration, $s = j$ and $j = n - 1$. Therefore, $j \in \mathbb{N}$, and s for now equals $\sum_{i=j+1}^{n} i$, or j in this case.

Prove that if I holds and the loop condition fails, then the post-condition holds.

Assuming that I holds and the loop condition fails, that means the loop ends. Therefore, the post-condition will hold.

Question 2

Consider the following algorithm, which takes as input $a, d \in \mathbb{N}^+$:

Initialize variables $r = a$ and $q = 0$.

While $r \geq d$, do the following:

Decrement r by d

Increment q by 1

Output the values of q and r.

$$\text{Pre-condition: } a, d \in \mathbb{N}^+$$
$$\text{Post-condition: } a = dq + r \ \wedge 0 \leq r < d$$
$$\text{Loop Invariant } I : a = dq + r \ \wedge \ r, d \in \mathbb{N}$$

Prove the pre-condition implies I.

Since both a and d are both positive integers, this implies that r and d are both natural numbers, since $r = a$.

Prove that if I and the loop condition both hold right before the iteration of the loop, then I holds right after the iteration.

Before the iteration, $r = a$, $q = 0$, and $d = n$.

After the iteration, $r = a - d$, $q = 1$, and $d = n$. We can convert $r = a - d$ to $a = d + r$, and since $q = 1$, we can say $a = dq + r$.

Prove that if I holds and the loop condition fails, then the post-condition holds.

If the loop condition fails, that means $r < d$, meeting the post condition. Additionally, if I holds, the post-condition is identical to I, meaning both parts of the post-condition are met.

Question 3

Use structural induction to prove that the number of vertices in any full binary tree is at most $2^{h(T)+1} - 1$.

BASIS

When a single node is the only node in the tree, $n(T) = 1$ and $h(T) = 0$.

Therefore, $n(T) = 2^1 - 1 = 1$

RECURSIVELY

$n(T)$ can be defined as $1 + n(T_2) + n(T_1)$ whenever T_1 and T_2 are a full binary tree

It can be assumed that $n(T_1) \leq 2^{h(T_1)+1} - 1$

Additionally, also $n(T_2) \leq 2^{h(T_2)+1} - 1$.

NOW,

$$n(T) = 1 + n(T_2) + n(T_1) \leq 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1)$$

$$n(T) \leq 2.2^{h(t)} - 1$$

$$n(T) \leq 2^{h(t)+1} - 1$$

which means the most number of nodes in a full binary tree will be $2^{h(t)+1} - 1$.

Question 4

In your solution to this question, you must use the set $B*$ of binary strings and the recursive definition of concatenation of binary strings that we gave in lecture, and you must state explicitly when you use those definitions.

a. Prove by structural induction that for all binary strings x, $\lambda x = x$.

There are only two cases for this:

1. If $x = \lambda$ then $x \times \lambda = \lambda \times \lambda = \lambda$
2. If $x \neq \lambda$ then $x \times \lambda = a \times w \times \lambda = a \times w = x$.

Therefore, in both cases $x \times \lambda = x$.

b. Consider the function reverse. Give a recursive definition for reverse.

For any string x and y, $x \cdot y$ is the combination of x and y. Therefore,

$$x \times y = \begin{cases} y \text{ if } x = \in \\ a(w, y) \text{ if } x = a \cdot w \end{cases}$$

Therefore, `reverse(x)` =

$$\begin{cases} \in \text{ if } x = \in \\ \text{reverse(w)}, a \text{ if } x = a \cdot w \end{cases}$$

c. Prove that `reverse(xy) = reverse(y)reverse(x)`.

We can prove this by induction.

BASE CASE

reverse(n) = n

RECURSIVE STEP

reverse(wa) = a * reverse(w).

Prove $\forall y$, $x \in \sum^*$, reverse(xy) = reverse(y) * reverse(x).

Let $P(y)$ be the property

$$\forall \in \sum^*, \text{ reverse}(xy) = reverse(y) * reverse(x)$$

Base case:

$y = e$.

reverse(x * e) = reverse(x).

reverse(e) * reverse(x) = reverse(x).

Thus, $P(y)$ holds.