Quiz Section 8: Arrays

Task 1 – Better Get Proving [14 pts]

The function replace : $(\text{List}, y, z) \to \text{List}$ is defined by

$$\begin{aligned} \text{replace}(\text{nil}, y, z) &:= \text{nil} \\ \text{replace}(x :: L, y, z) &:= z :: \text{replace}(L, y, z) \quad \text{if } x = y \\ \text{replace}(x :: L, y, z) &:= x :: \text{replace}(L, y, z) \quad \text{if } x \neq y \end{aligned}$$

In this problem, we will prove that replace works the same way at the end of the list that it does at the front of the list, i.e.:

$$\begin{aligned} \text{replace}(L +\!\!+ [x], y, z) &= \text{replace}(L, y, z) +\!\!+ [z] \quad \text{if } x = y \\ \text{replace}(L +\!\!+ [x], y, z) &= \text{replace}(L, y, z) +\!\!+ [x] \quad \text{if } x \neq y \end{aligned}$$

a) Explain, in your own words, why the following statement, if proven, would imply the one above:

$$\text{replace}(L +\!\!+ [x], y, z) = \text{replace}(L, y, z) +\!\!+ \text{replace}([x], y, z)$$

b) Explain, in your own words, why the following claim, if proven, would imply the one from part (a):

$$\text{replace}(L +\!\!+ R, y, z) = \text{replace}(L, y, z) +\!\!+ \text{replace}(R, y, z)$$

c) Prove the claim from part (b) by induction <u>on L</u>.

Task 2 – Jumping Through Loops [16 pts]

In this problem, you will implement the following function:

```
/**
 * Writes over each copy of y in A with the value z.
 * @param A .. y .. z ..
 * @modifies A
 * @effects A = replace(A_0, y, z)
 */
public void replace(int[] A, int y, int z) { .. }
```

With each loop invariant below, fill in the missing parts of the code to make it correct with the **given invariant**. (If the code works correctly with some other invariant, it is not correct.)

a)

```
int i = __________

// Inv: A[.. i] = A_0[.. i] and A[i+1 ..] = replace(A_0[i+1 ..], y, z)
while (__________) {



}
```

b)

```
int i = __________

// Inv: A[.. i-1] = replace(A_0[.. i-1], y, z) and A[i ..] = A_0[i ..]
while (__________) {



}
```

Task 3 – Rally the Loops **[15 pts]**

Recall the function remove : (List, y) $\to$ List defined as follows:

$$\begin{aligned} \text{remove}(\text{nil}, y) &:= \text{nil} \\ \text{remove}(x :: L, y) &:= \text{remove}(L, y) && \text{if } x = y \\ \text{remove}(x :: L, y) &:= x :: \text{remove}(L, y) && \text{if } x \neq y \end{aligned}$$

It is possible to prove, as the same manner as we did in Task 1, that the following holds:

$$\begin{aligned} \text{remove}(L + [x], y) &= \text{remove}(L, y) && \text{if } x = y \\ \text{remove}(L + [x], y) &= \text{remove}(L, y) + [x] && \text{if } x \neq y \end{aligned}$$

You can use these facts below without proof. Refer to them as "Lemma 2".

In this problem, we will check the correctness of the following code that implements remove. Specifically, it writes remove(A, y) into some prefix of the array, $A[..\, j-1]$, and returns j.

```
int i = 0;
int j = 0;
{{ P1: ________________________________ }}
{{ Inv : A[..j-1] = remove(A[..i-1], y) and A[i..] = A0[i..] }}
while (i != A.length) {
  if (A[i] == y) {
      {{ P2: ____________________________ }}
      {{ Q2: ____________________________ }}
  } else {
      {{ P3: ____________________________ }}
      {{ Q3: ____________________________ }}
      A[j] = A[i];
      j = j + 1;
  }
  i = i + 1;
}
{{ P4: ________________________________ }}
{{ Post: A[..j-1] = remove(A, y) }}
return j;
```

a) Fill in P_1 using forward reasoning and then prove that P_1 implies Inv.

b) Fill in P_4 using forward reasoning and then prove that P_4 implies the postcondition.

c) Fill in P_2 using forward reasoning and Q_2 using backward. Then, prove that P_2 implies Q_2.

d) Fill in P_3 using forward reasoning and Q_3 using backward. Then, prove that P_3 implies Q_3.