Assignment 5: Proving Programs Correct

Due: 11:00am, Wednesday, October 10

- Emails about this assignment should be directed to **cs81help@cs.hmc.edu** .
- The usual collaboration rules apply. You may *discuss* an exercise with any other student(s) currently taking CS 81 as long as:
  - You contribute equally;
  - You come away from this discussion only with *understanding in your head* — no written materials or computer notes may be retained;
  - Your submission is authored solely by you, on a separate occasion.
- You should refer only to materials from this semester of CS 81 (lecture notes, handouts, textbooks, grutors, profs, etc.).
- Bring a writeup/printout to class on the due date. Illegible answers will get no credit.
- Make sure your submission includes your name!

1 Maximum of a Function [30 points]

Consider the following code for finding the maximum of a function f on the first $N \geqslant 1$ natural numbers).

$\{ \; N \; \geqslant \; 1 \; \}$

```
j = 1
m = f(0)
while (j != N):
   if m > f(j):
       j = j + 1
   else:
       m = f(j)
       j = j + 1
```

$\{ \quad m = \max_{k \in 0..N-1} f(k) \; \}$

1. [10 points] Specify a suitable loop invariant I. (It should be true before the loop starts, remain true after every iteration, and imply the desired postcondition when the loop has terminated.)

2. [10 points] Use the proof rules of Hoare Logic to prove the code partially correct. In your "implied" steps, you can make any mathematically correct deduction and do not need to show the proof of that deduction.

3. [10 points] Specify a loop variant that could be used to prove termination (and hence total correctness). You do not need to supply such a proof.

Hint: my partial correctness proof had the following form, where I is my loop invariant.

```
{ N ⩾ 1 }                                          PRECONDITION
{ ... }                                            IMPLIED
j = 1
{ ... }                                            ASSIGNMENT
m = f(0)
{ I }                                              ASSIGNMENT
while (j != N):
    { I ∧ j ≠ N }                                    ASSUMPTION
    if m > f(j):
        { I ∧ j ≠ N ∧ m > f(j) }                       ASSUMPTION
        { ... }                                        IMPLIED
        j = j + 1
        { I }                                          ASSIGNMENT
    else:
        { I ∧ j ≠ N ∧ ¬(m > f(j)) }                    ASSUMPTION
        { ... }                                        IMPLIED
        m = f(j)
        { ... }                                        ASSIGNMENT
        j = j + 1
        { I }                                          ASSIGNMENT
    { I }                                            IF
{ I ∧ ¬(j ≠ N) }                                   WHILE
{ m = max_{k∈0..N−1} f(k) }                      IMPLIED
```

2 Remainder via repeated subtraction [35 points]

The following code computes $a \bmod d$ (where $a \geqslant 0$ and $d > 0$) via repeated subtraction, and puts the final result in r. It tries to speed things up (e.g., when a is much larger than d) by subtracting larger and larger multiples of d where possible.

$\{\ a \geqslant 0 \ \wedge \ d > 0\ \}$

```
r = a
while (r >= d):
   g = d
   while (r >= g):
      r = r - g
      g = g + g
```

$\{\ r \ = \ a \bmod d\ \}$

1. [10 points] Specify a suitable invariant I_1 for the outer **`while`**. (It should be true before the loop starts, remain true after every iteration, and imply the desired postcondition when the loop has terminated.)

2. [10 points] Specify a suitable invariant I_2 for the inner **`while`**. (It should be true before the inner loop starts, remain true after every iteration of the inner loop, and imply I_1 when the inner loop has terminated. It probably should mention **`g`**.)

3. [10 points] Use the proof rules of Hoare Logic to prove the code partially correct. In your "implied" steps, you can make any mathematically correct deduction and do not need to show the proof of that deduction.

4. [5 points] Specify a variant for each loop. (Again, no proof is required.)

Hint: again, my proof skeleton.

$\{\ a \geqslant 0\ \wedge\ d > 0\ \}$ **PRECONDITION**
$\{\ \ldots\ \}$ **IMPLIED**
`r = a`
$\{\ I_1\ \}$ **ASSIGNMENT**
`while (r >= d):`
  $\{\ I_1\ \wedge\ r \geqslant d\ \}$ **ASSUMPTION**
  $\{\ \ldots\ \}$ **IMPLIED**
  `g = d`
  $\{\ I_2\ \}$ **ASSIGNMENT**
  `while (r >= g):`
    $\{\ I_2\ \wedge\ r \geqslant g\ \}$ **ASSUMPTION**
    $\{\ \ldots\ \}$ **IMPLIED**
    `r = r - g`
    $\{\ \ldots\ \}$ **ASSIGNMENT**
    `g = g + g`
    $\{\ I_2\ \}$ **ASSIGNMENT**
  $\{\ I_2\ \wedge\ \neg(r \geqslant g)\ \}$ **WHILE**
  $\{\ I_1\ \}$ **IMPLIED**

$\{\ I_1\ \wedge\ \neg(r \geqslant d)\ \}$ **WHILE**
$\{\ r\ =\ a \bmod d\ \}$ **IMPLIED**

Exponentiation [35 points]

Finally, here is some code that sets z to be a^b, where a and b are nonnegative integers. For efficiency, it uses repeated squaring. (Recall that `%` is the standard way of writing the "modulo" or "remainder" operator in code.)

$\{\ a > 0\ \wedge\ b \geqslant 0\ \}$

```
x = a
y = b
z = 1
while (y != 0):
    while (y % 2 == 0):
        x = x * x
        y = y / 2
    y = y - 1
    z = z * x
```

$\{\ z\ =\ a^b\ \}$

1. [10 points] Specify a suitable invariant I_1 for the outer **`while`**. (It should be true before the loop starts, remain true after every iteration, and imply the desired postcondition when the loop has terminated.)

2. [10 points] Specify a suitable invariant I_2 for the inner **`while`**. (It should be true before the inner loop starts, remain true after every iteration of the inner loop, and imply whatever is necessary when the inner loop has terminated.)

3. [10 points] Use the proof rules of Hoare Logic to prove the code partially correct. In your "implied" steps, you can make any mathematically correct deduction and do not need to show the proof of that deduction.

4. [5 points] Specify a variant for each loop. (Again, no proof is required.)