# Assignment 6: Temporal and Hoare Logics

Due: 1:15pm, Tuesday, October 5

- Emails about this assignment should be directed to cs81help@cs.hmc.edu .
- Grutor office hours in Platt are Sundays 8-10pm and Mondays 9-11pm; Prof. Stone's office hours in Olin 1251 are MW 4-5pm and TR 3-4pm and by appointment.
- 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!

---

**Note: The material covered by the relevant lectures also appears in much more detail in Sections 3.4 and 4.2–4.3 of Huth and Ryan.**

## 1 Temporal Logic

| | Eventually True | Invariably True |
|---|---|---|
| Some Path | EF $p$<br>Possibly | EG $p$<br>Potentially Always |
| All Paths | AF $p$<br>Inevitably | AG $p$<br>Invariably |

Consider the possible execution paths in the following piece of code:

```
int a[100] = { 0, 0, 0, ..., 0 };  // Array is initally zeroed

while (true) {
   int k = user_input();   // Assume this user input will be a number
                           // between 0 and 99 inclusive!

   if (a[k] == 1) {
       for (int i = 0; i < 100; ++i)
           a[i] = 0;
   } else {
       a[k] = 1;
   }
}
```

Which of these logical propositions are true about all possible executions (starting at the beginning)? Explain your answer; you do not need to provide a full formal proof, but your explanation should be clear and convincing.

1. $\mathrm{AG}\,(\mathsf{a}[42] = 0)$
2. $\mathrm{AG}\left(\sum_{j=0}^{j<100} \mathsf{a}[j] < 100\right)$
3. $\mathrm{EF}\,(\mathsf{a}[42] = 1)$
4. $\mathrm{AF}\,(\mathsf{a}[42] = 1)$
5. $\mathrm{EG}\,(\mathsf{a}[42] = 0)$
6. $\mathrm{AG}\,(\mathrm{AF}\,(\mathsf{a}[42] = 0))$
7. $\mathrm{AG}\,(\mathrm{EF}\,(\mathsf{a}[42] = 1))$
8. $\mathrm{EF}\,(\mathrm{AG}\,(\mathsf{a}[42] = 0))$

# 2 Hoare Logic

1. The well known "3n+1 problem" (more formally known as the Collatz Conjecture) asks whether a particular process will always terminate. Given any number, transform it using the following sequence of operations:

   - If the number is even, divide by 2.
   - If the number is odd, multiply by 3 and add 1.
   - Stop if you reach the number 1; otherwise, repeat from the top.

   For example, we start with 6, then we proceed to 3, 10, 5, 16, 8, 4, 2, 1, and stop.

   Although every starting point examined so far has eventually reached 1, nobody has proved this will always happen. Therefore, no one knows how to prove *total* correctness of the following code:

   $\{ n \geqslant 1 \}$

   ```
   while (n != 1) {
     if (n % 2 == 0)
       n = n / 2;
     else
       n = 3 * n + 1
   }
   ```

   $\{ n = 1 \}$

   However, demonstrating the partial correctness is really easy. Prove that this code is partially correct.

2. Prove *total* correctness of the following factorial-computing code, by providing a variant and a Hoare-logic tableaux. Be clear about what loop invariant you are using; you can use (correct) logical implications in the IMPLIED rule without proving them.

$\{ x \geqslant 0 \}$

```
a = x;
y = 1;
while (a > 0) {
  y = y * a;
  a = a - 1;
}
```

$\{ y = x! \}$

3. Assume we are given an array a of N integers and a value V to search for in that array.

   [We are cheating a bit because the Hoare Logic rules for arrays are more complex than what we have seen in class. However, since the arrays in question never change, a "naive" approach will still work.]

   If we can make no assumptions about the array, then linear search (checking each element of the array in turn) is the best that we can do.

   Provide a short code fragment implementing linear search, and prove total correctness *under the precondition that the value being searched for is in the array*

   - You will need to provide an appropriate precondition, postcondition, and loop invariant. Make these explicit.
   - Provide a "proof tableaux" showing the partial correctness of your code (relative to your precondition and postcondition). As usual, you do not need to prove the purely logical/mathematical implications that arise in the IMPLIED rule.
   - Provide a variant to show termination, along with a brief but convincing argument that this variant is both nonnegative and strictly decreasing.