Assignment 4: Arrays
15-414/15-424 Bug Catching: Automated Program Verification

Due: **11:59pm**, Thursday 9/28/17
Total Points: 50

1. **Spec a partition (5 points)** Suppose you are given the following code, which partitions elements in the range $[l, u]$ of an array a of length n on a given index p where $0 \leqslant p < n$. That is, after the code runs all elements between indices l and u, where $l \leqslant p < u$, that are at most $a(p)$ are on the "left" of a and all elements greater than $a(p)$ are on the "right".

```
v := a(p);
a(p) := a(u);
a(u) := v;
i := l - 1;
j := l;
while(j < u) {
  if(a(j) <= v) {
    i := i + 1;
    t := a(i);
    a(i) := a(j);
    a(j) := t;
  }
  j := j + 1;
}
t := a(i+1);
a(i+1) := a(u);
a(u) := t;
```

   Your first task is to formalize the correctness of this code according to the informal description above by defining `pre` and `post`. Ultimately, you would like to define these so that the DL formula `pre` $\rightarrow [\alpha]$`post` is valid exactly when the code behaves as described above (where α is the above code). If you find the given description ambiguous in any way, then be sure to explicitly state your interpretation of the functionality and any resulting assumptions you made.

2. **Find an invariant (5 points)** Now that you have specified the behavior of the code from problem 1, write a loop invariant that satisfies all of the conditions necessary to invoke the loop rule, leading to a proof that `pre` $\rightarrow [\alpha]$`post` is valid. You do not need to formally prove this, but you do need to give a brief informal explanation as to why your invariant is correct.

3. **Sum and max (10 points)** Let α correspond to the following program:

```
while(i < n) {
  if(m < a(i)) {
    m := a(i);
  }
  s := s + a(i);
  i := i + 1;
}
```

   Use the axioms of dynamic logic to conduct a sequent calculus proof that the following formula is valid:

$$s = 0, m = 0, i = 0 \vdash (\forall j. 0 \leqslant j < n \rightarrow a(j) \geqslant 0) \rightarrow [\alpha] s \leqslant n \cdot m$$

   Be sure to clearly state your loop invariant to help your graders understand your solution.

4. **Spec an inversion (5 points)** Suppose that an array a is an injection: distinct indices map to distinct elements. Furthermore, we assume that a is defined on all indices i such that $0 \leqslant i < n$, and that it only maps to values in this range as well. We want to write a program that inverts a into a second array b, so that if a for example starts out as (for $n = 4$):

$$[3, 1, 0, 2]$$

   Then after the code runs, b has the value:

$$[2, 1, 3, 0]$$

   Your task is to write a specification for this program by giving `pre` and `post`. As in problem 1, ff you find this description ambiguous, then explicitly state your interpretation of the functionality and any resulting assumptions that you make.

5. **Implement it (5 points)** Now that you have specified the behavior from problem 4, write a program to implement the functionality. Then, write a loop invariant that will allow you to prove its correctness with respect to your spec. *Hint: your program can and should be very short!*

6. **Prove your inversion (10 points)** Using the axioms of dynamic logic, conduct a sequent calculus proof that your implementation from problem 5 is correct.

7. **Array terms (5 points)** Using the read-over-write axioms, conduct a proof in the sequent calculus of the following formula.

$$a\{i \mapsto e\}(j) = e \rightarrow i = j \vee a[j] = e$$

8. **Soundness of array updates (5 points)** In lecture we discussed an axiom for reasoning about array updates:

$$([:=]_{()}) \quad [a(e) := \tilde{e}]p(a) \leftrightarrow p(a\{e \mapsto \tilde{e}\})$$

   Use the following semantics of array update to prove the validity of the formula above:

$$[\![a(e) := \tilde{e}]\!] = \{(\omega, \nu) : \omega = \nu \text{ except } \nu[\![a]\!] = \omega(a)\{\omega(a)[\omega[\![e]\!] \mapsto \omega[\![\tilde{e}]\!]\}\}$$