Computer Science 81, Spring 2013

Assignment 6

Due Wed. March 6, 2012, 11:59 PM

1. [40 points] Prove the **total** correctness of the Python program below, with respect to the assumption and expectation indicated.

```
# input is b (which never changes) and n (initial value n0)
# all values are integer throughout
# assumption: n == n0 & n0 >= 0 & b > 0

s = b
r = 1
while n > 0:
    if n % 2 == 1:   ### % is the modulus (remainder) function
        r = r * s
    n = n // 2       ### // is integer (flooring) division
    s = s * s

# expectation: r == pow(b, n0)
# pow is the power function, raising b to the power n0
```

The first task is to determine how this program works. Devising an appropriate loop invariant should help. Note that the invariant will usually involve *all* of the variables used in the loop. To prove termination, you will need to establish a variant as well.

2. [50 points] The Python program below is supposed to perform binary search: given a value and a list already sorted in increasing order, it is supposed to return an index at which the value occurs in the list, or -1 if the value does not occur therein.
    a. Create a specification for binary_search, in the form of an assumption and an expectation stated in predicate logic.
    b. Devise an invariant assertion for the while loop.
    c. Check the invariant empirically by inserting it at appropriate points in the program using `assert` statements, then running the program.

```
found = -1
low = 0
high = len(A)-1
while (found == -1) & (low <= high):
    mid = (low+high)//2
    if A[mid] == value:
        found = mid
    else:
        if A[mid] > value:
            high = mid-1
        else:
            low = mid+1
```

d. Propagate the invariant backward through the body of the while loop using Hoare logic methods, creating additional assertions, and check them incrementally by running the program.

e. Try to show that the invariant with the test condition imply the assertion formed by propagating backward through the loop body. This may be challenging, as the expressions grow larger as they are propagated backward, but it should be possible in principle.

By inspection of the code, the array A is never modified, so this fact can be left implicit to simplify the argument.

Suggestions: (I): Use Python's `in` operator to express some of your assertions. For example:

```
v in A[low:high]
```

means that `v` occurs in array `A` at a position between `low` and `high-1`.

(II): Use Python's `if` *expression* (not statement):

val1 `if` P `else` *val2*

which evaluates to *val1* if P evaluates to True, and evaluates to *val2* otherwise. This can be handy for dealing with the conditional rule in Hoare logic, as these expressions can be nested arbitrarily, as nested implications. (This is like `P ? val1 : val2` in Java or C++).

You may ask for additional hints on this one. You might ask me to comment on your invariant.

Extra Credit [50 points] Prove the program using JAPE.

3. [10 points] Establish the satisfiability or unsatisfiability of the following sets of clauses by the *resolution* method. If a set is satisfiable, give a valuation that shows this.

   a. {p ∨ q, p ∨ ¬r, ¬p ∨ r, ¬p ∨ ¬q, ¬q ∨ r, q ∨ ¬r }

   b. {p ∨ q, ¬p ∨ r, ¬p ∨ ¬q, ¬q ∨ r, q ∨ ¬r }