CS 598MP: Software Verification, Fall 2010
Problem Set 1 (due Thursday, October 7th, 3:30pm)

The deadline is firm! Turn in your homework *before class begins, in class*; homework will not be accepted after the class begins. You can also hand over the homework during normal office hours anytime before the deadline at Elaine Wilson's office (3229 Siebel) (slide it under the door if the door is locked).

1. Floyd-Hoare style proofs (20 pts)

   The following program purports to compute the square of a given integer n (not necessarily postive). Prove the program's partial correctness (i.e. that if it halts, it computes the square of n, for any input n), by giving a Floyd-style proof. Do this by giving an inductive invariant (an invariant at every point in the program). Also, give the set of all individual *verification conditions* (which must be valid, of course) that proves that the invariants are really invariant. You can use either weakest pre-conditions or strongest post-conditions to formulate the verification conditions.

   Finally, write down a proof of termination, assuming the pre-condition on the input that $n > 0$, by giving appropriate ranking functions that map states to a well-founded ordering, and argue why this proves termination.

```
int i, j;
i := 1;
j := 1;
while (j != n) {
  i := i + 2*j + 1;
  j := j+1;
}
return j;
```

2. Fixed-points: (8 pts)

   Let S be a finite set. Fix the $\subseteq$ ordering on 2^S.
   Let $f : 2^S \to 2^S$ be a monotonic function.

   Consider the following sequence of sets:
   $T_0 = S$
   $T_1 = f(T_0)$
   $T_2 = f(T_1) \ldots$

   Prove that:

(a) The sequence must converge; i.e. for some n, $T_n = T_{n+1}$.

(b) Show that the set it converges to is the *greatest fixed-point* of the function f.

3. Termination: (12 pts)
   Consider the following program:

```
int a,b;
a := 1000; b := 0;
while (a !=0 || b != 0) {
  if (b==0) {
    a := a-1;
    b := f() }
  else {
    b := b-1;
}
```

   In the above, $f()$ is a function that returns arbitrary positive numbers each time it is called (it need not return the same number on two successive invocations). Think of $f()$ as, say, a function that returns a number from the environment (input). More formally, all that we know is that $f()$ returns a value greater than 0.

   Prove that the above code terminates always by giving a proof based on ranking functions.

4. Symbolic testing: (10 pts)

   Consider the following straight-line program:

```
x:=x+10;
y:=x*2;
assume(y>24);
x:=y+y;
assume(2x-y>100);
```

   Write down the path-constraint corresponding to the above path, as a pure logical arithmetical formula φ (the satisfiability of φ would mean the path is feasible).

   Either using Z3 or Yices or by manual reasoning, derive a satisfiable valuation that traces this path. Also, using the PEX-for-fun website, write a corresponding $C\#$ program and use it to find an initial value for x and y that traces this path (include a print-out of the PEX output).