Program Analyis and Verification
Assignment 5
Hoare logic and Type systems
Due date: Dec. 6th

Problem 1.
Consider the program P below which performs division using repeated subtraction:

```
q := 0;
r := x;
while (r >= y) {
  r := r - y;
  q := q + 1;
}
```

Give a proof of the following assertion, using (a) Floyd-style inductive annotation and (b) Hoare logic.

$$\{x \geq 0 \wedge y \geq 0\}\ P\ \{q \cdot y + r = x \wedge 0 \leq r < y\}.$$

Problem 2.
Consider a variant of the while statement of the form "`do S while(b)`".

1. Give the weakest precondition for

```
do
  x := x+1;
while (x <= 10)
```

2. Describe $WP(do\ S while(b))$ in terms of $WP(while(b)\ do\ S)$.

Problem 3.
Consider the following lambda term:

$$\lambda x.\lambda y.((\lambda z.(x\,(z+1)))((\lambda w.w+1)\,y))$$

1. Show the term above with suitable type annotations given to all variables introduced in λ-abstractions, such that the term becomes well-typed. Assume that all primitive values in this problem are integers.

2. Depict a run of the TypeCheck algorithm on the annotated term that you will show above. Since the algorithm is recursive, the run consists of a set of invocations. To each invocation you need to assign an identification number for convenience of presentation. The "root" invocation will have identification number zero, the next invocation (corresponding to the '$\lambda y.((\lambda z. \ldots y))$' subterm)

would have number 1, etc. For each invocation, you will write (a) its identification number, (b) the identification number of its parent invocation (this won't exist for the root invocation), (c) the two arguments received into the invocation (assume that the root invocation is given an empty environment), and (d) the type returned by the invocation. You can assume the algorithm has been appropriately extended to handle arithmetic operations. You need not specify the details of the intermediate computations within each invocation.

3. Show a proof tree for the annotated term that you will show for Part 1 above, treating the type returned by TypeCheck in your solution to Part 2 above as the type of the term. For each node in the proof tree, indicate the corresponding invocation number in your solution to Part 2.

Problem 4. Prove the preservation and progress theorems for Simply Typed Lambda Calculus (STLC) terms rooted at the function-application operator, for terms of *height* $n+1$. The height of a term is the maximum nesting depth of subterms of the term. You can assume that the progress and preservation theorems hold for all terms of height at most n.

Note, in your proofs, you may need to invoke some of the "inversion lemmas" and "canonical forms" lemmas, which are mentioned in the slides. In such cases, state clearly which of these lemmas you are invoking, and where. You don't need to prove these lemmas.