Programming Languages

Fall 2008

May 7, 2008

NOTE: this exam is open books and open notes.

Question 1 [40 Points]

Let us consider the simple imperative language that we studied in the context of axiomatic semantics (simple language with assignments and while loops). Let us introduce in the language the ability to define parameter-less procedures, with the following syntax:

```
<Program>    ::=  <Procedures> <Command>
<Procedures> ::=  <Procedure> ; <Procedures>
               |    epsilon
<Procedure>  ::= procedure <Id> <Command>
<Command>    ::= <Id> = <Expression>
               |   if <Expression> then <Command> else <Command>
               |   while <Expression> do <Command>
               |   call <Id>
               |   <Command> ; <Command>
```

Perform the following tasks:

15 Points Develop an *axiomatic semantics* rule (in the style of Hoare's axioms) for the procedure call—i.e., something of the type:

$$\frac{\textbf{...}}{\{p\}\ \textbf{call}\ Id\ \{q\}}$$

1. (4 Points) Consider the following code fragment; provide meaningful preconditions and postconditions.

```
procedure foo x = x*2; y=y-1 ;

{   PRECONDITION  }
x=1; y=10;
while (y > 0)
  call foo
{   POSTCONDITION  }
```

2. (6 Points) Provide a proof of correctness for the code fragment with respect to the precondition and postcondition you have developed.

   NOTE: you need to develop the proof in the usual manner and only for the part of the code enclosed between the precondition and the postcondition.

15 Points Provide the sequents that describe the *Operational Semantics* of the procedure definition and procedure call.

Question 2 [60 Points]

Let us consider the following syntax for an imperative language:

```
<Program>    ::= main () <Command>
<Command>    ::= <Command> ; <Command>
              |   <Id> = <Expression>
              |   *<Id> = <Expression>
              |   while <Expression> <Command>
              |   if <Expression> <Command> else <Command>
              |   <Id> = malloc <Number>
<Expression>::= <Number>
              |   <Id>
              |   & <Id>
              |   * (<Id> + <Number>)
              |   <Expression> + <Expression>
```

The language allows the use of pointers and pointer variables and the allocation of memory. Note that the expression allows some basic pointer arithmetic. Provide the denotational semantics for this language; explicitly indicate any reasonable assumptions you are making in developing your solution. You are required to include in your semantics checks to guarantee correctness of pointer accesses (i.e., pointers dereference to an existing variable or to an area of memory that has been allocated).