Programming Languages
Qualifying Exam

Spring 2012
New Mexico State University

November 29, 2011

NOTE: this exam is open book and open notes.

Question 1 [30 Points]

Consider the following program in the flow-chart language:

$1:$ *start* $n > 0 \wedge \exists A(n = 2 * A + 1)$
$2:$ $d = 0$
$3:$ $c = 0$
$4:$ *assert* q
$5:$ **if** $c = n + 1$ **goto** 9
$6:$ $c = c + 2$
$7:$ $d = d + 1$
$8:$ **goto** 4
$9:$ $s = 0$
$10:$ *assert* r
$11:$ **if** $d = 0$ **goto** 13
$12:$ $s = s + n$
$13:$ $d = d - 1$
$14:$ **goto** 10
$15:$ *stop* $s = \sum_{i=1}^{n} i$

- develop the most appropriate assertions for lines 4 and 8;

- using Floyd's method, prove partial and total correctness of the program.

Question 2 [20 Points]

Consider the following simple syntax for commands in an imperative language:

```
<command>   ::= <id> := <expression>
      |         if <expression> then <command> else <command> endif
      |         while <expression> do <command> end while
      |         <id> := read_file(<id>)
      |         write_file(<expression>, <id>)
```

The novelty here is the read and write operations; the read gets the next element from the given sequential file (e.g., **read_file(F)** will return the next element from the file **F**); the write will place the value of the expression at the end of the given file (e.g., **write_file(v,F)**) will write the value **v** at the end of the file **F**.

Provide the **either** the (big-step) operational semantics for the commands **or** the denotational semantics—making sure to properly characterize the state; your semantics should allow arbitrary files to be used.

Question 3 [50 Points]

Let us consider the following syntax for an imperative language (only the parts of interest are presented, the others are the standard ones seen in class):

```
<values_list>  ::=  <number>
    |               <number> , <values_list>
<command>   ::= <identifier> = <expression>
    |      <command> ; <command>
    |      if <expression> then <command>
    |      search <identifier> in <values_list> causing <expression>
                                                 after <command>
            endsearch
    |      fail
```

The novelty is the search command; for example

```
search x in -10, 10,100,1000 causing r == 2 after
      if x < 0 then fail ;
      r = log10 (x)
endsearch
```

the command explores the different assignments of the identifier (e.g., x in the example) with values drawn from the given list of values (e.g., $-10, 10, 100, 1000$ in the example). The goal is to find an assignment for the identifier such that the expression evaluates to true after the execution of the given command. In the example, the value of r should be 2 after the execution of the given command. The command search should automatically try alternative values until one meeting the condition is found. The fail command will stop the command and force the search to explicitly move to another value.

Provide the complete denotational semantics for the commands and the values list (you can assume the standard semantics for the other components of the language).