Spring 2019 Ph.D. Qualifying Exam – Languages

Your Test ID Number: _____

Instructions

Write your test id number above and on each page of your answers. This exam is closed book and closed notes. You must pass both parts to pass the test.

Part 1: CS 536 [50 points]

(1) (6 points) For the nondeterministic program below, you're given q_1 and q_2. What are the most general w_1, w_2, p, and q that make this triple correct?

{p} if $w_1 \rightarrow S_1$ {q_1} □ $w_2 \rightarrow S_2$ {q_2} fi {q}

(2) (8 points) Calculate the indicated weakest precondition below, following the definition of wp. Show your work. (In particular, show the results of all substitutions; don't leave them in predicate[expr/var] form.) Don't worry about possible out-of-range indexes for b. You may logically simplify your predicates, during and at the end of the calculation.

wp(b[x] := c, b[b[x]] > 0)

(3) (8 points) Calculate the indicated strongest postcondition below, following the definition of sp. As in the previous problem, show your work; you may logically simplify your predicates.

sp(x * y < z, x := x+y; x := x–z)

(4) (12 points) Give definitions for p_1, p_2, p_1, and q_2 so that the parallel program outline below is interference-free and deadlock-free. List the interference-freedom and deadlock-freedom tests you find.

{ $x > 1 \wedge y > 1$ }

[{ p_1 } await y > 1 then x := x–1 end { q_1 }

|| { p_2 } await x > 0 then y := y^2 end { q_2 }

]

{ $x \neq 0 \wedge y \neq 0$ }

(continued)

(5) (16 points) The following program calculates the integer $\log_2(x_o)$ by terminating with $2^r \le x_o < 2^{(r+1)}$. It is incompletely annotated. Complete the definition for the invariant p and define $p_1 – p_4$ so that you get a proof outline for partial correctness.

Don't alter the program or the conditions already given (except for completing them, of course). You may add more conditions { .... } if you like. You can use substitution notation $p[e/v]$ but show the results of the substitutions somewhere.

$\{x = x_o \ge 0 \wedge r = 0\}$

{inv p} $p \equiv 1 \le x_o/2^r < ....$ // Hint: add x to p

while $x \ge 2$ do

$\{ p_1 \}$

$\{ p_2 \}$

x := x / 2;

r := r + 1

$\{ p_3 \}$

od

$\{ p_4 \}$

$\{2^r \le x_o < 2^{(r+1)}\}$

Spring 2019 Ph.D. Qualifying Exam – Languages

Part 2: CS 440 [50 points]

Question 1: Languages [25 points]

For each pair of terminologies in the following. Give one advantage and one disadvantage of the first term as compared to the second term. For example, in (1a) give one advantage and one disadvantage of an interpreter over a compiler.

(1a) Interpreter vs compiler

(1b) Exception type checking (as in Java) vs no exception type checking (as in C#)

(1c) Call by value vs call by reference

(1d) Compile into byte code (as in Java) vs not

(1e) Recursion vs iteration (in writing a program to walk a tree)

Question 2: Exception Handling [10 points]

Most programming languages (e.g. C++ and Java) use a "termination model" to specify control flow after an exception is raised in the middle of a try block. Statements in the rest of the try block will be skipped; control goes directly to the catch block.

(2a) [3 points] What is wrong with the following fragment of code?

```
try {
      socket.close();
      connection.close ();
      inputStream.close ();
} catch (Exception e) {
      …     //exception handling code
}
```

(2b) [3 points] Rewrite the code fragment in a) to correct its defects.

(2c) [4 points] Are you happy with your solution in (2b)? Why? Can you come up with a better language solution?

Question 3: Grammars [15 points]

One can use the following pair of BNF rules to define an if then else statement:

> *IfStatement* -> if (*Expression*) *Statement* | if (*Expression*) *Statement* else *Statement*
> *Statement -> Assignment* | *IfStatement*

(3a) [5 points) Use the following code fragment to show that the rules are ambiguous.

```
if (x < 0)
      if (y < 0) y = y + 1;
      else y = 0;
```

(3b) [5 points] Use your result of (3a) to describe the problem of ambiguous rules.

(3c) [5 points] Propose a method to solve this problem for the two rules.