Spring 2020 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) (8 points) For the nondeterministic program below, what relationships are needed between p, p1, and p2? q, q1, and q2? p1, S1, and q1? p2, S2, and q2? (Note you've been given postconditions for S1 and S2 but you need to find preconditions for S1 and S2.)

{p} if p1 -> S1 {q1} □ p2 -> S2 {q2} fi {q}

(2) (8 points) Find the weakest p such that {c < d ∧ p} b[c] := x { b[b[d]] > 0 }. Use logical simplifications on p as much as you can.

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

sp(p(c, d, e) , c := f(c, d); c := g(c, e)) [where c, d, e are variables]

Questions 4 and 5 refer to the following program:

```
{ p }
[     { p1 } await B1 then S1 end { q1 }
||    { p2 } await B2 then S2 end; { q2 } await B3 then S3 end { r2 }
]
{ q }
```

(4) (8 points) What are the interference-freedom tests for the program?

(5) (8 points) What are the deadlock-freedom tests for the program?

(6) (10 points) Complete most of the total correctness proof outline below by calculating predicates for p1, p2, p3, and p4. Use sp for p1 and wp for p3, but write out the results using pred [exp / var] substitution notation, not wp(…) or sp(…). Make the outline totally correct by including the bound wherever it's needed. You can assume all expressions evaluate without error.

```
{ p0 }              // p0 is given to us
x := e1;  y := e2;
{ p1 }              // Give sp using pred [ exp / var ] notation
{ inv p } { bd e }
while B do
      { p2 }
      { p3 }        // Give wp using pred [ exp / var ] notation
      x := e3; y := e4
      { p4 }
od
{ p ∧ ¬B }
```

Part 2: CS 440 [50 points]

(1) General (15 points) For each pair of programming language terminologies in the following, give the main difference between them and the main advantage of one over the other. For example, in the first pair, you will give the main difference between Java exception type checking and C# no exception type checking, an advantage of type checking and an advantage of no type checking.

a. (5 points) Exception type checking in Java vs no exception type checking in C#

b. (5 points) Pointers in C vs reference in Java.

c. (5 points) Recursion vs iteration

(2) Programming language design (20 points)

a. (5 points) Some programming languages provide the `resume` keyword to allow the programmer to specify the control flow that after an exception is thrown and handled, control flow resumes to the next statement in the `try` block. In the following example, if `statement2` throws an exception, after the `catch` block finishes, `statement3` will

get executed next. Java does not provide resume. You are asked to use Java to implement the same control flow and comment on whether Java should implement `resume`.

```
try (statement1;
     statement2;
     statement3;
} resume after catch (exception e) {...}
```

b. (5 points) Some programming languages provide the `retry` keyword to allow the programmer to specify the control flow that after an exception to allow the programmer to specify the control flow returns to the beginning of the `try` block. The usage of the `retry` keyword is illustrated in the following example. Java does not support `retry`. You are asked to use Java to implement the same control flow and comment on whether Java should implement `retry`.

(3) Grammar (10 points)

The following set of rules comes from the Java grammar file:

```
Statement ::= ...|IfThenStatement|IfThenElseStatement |
    StatementWithoutTrailingStatement

StatementWithoutTrailingStatement ::= ...|Block|
    EmptyStatement | ReturnStatement

StatementNoShortIf ::= ...|
    StatementWithoutTrailingSubstatement |
    IfThenElseStatementNoShortIf

IfThenStatement ::= if (Expression) Statement

IfThenElseStatement ::= if (Expression) StatementNoShortIF
    else Statement

IfThenElseStatementNoShortIf ::=
    if (Expression) StatementNoShortIf else
    StatementNoShortIf
```

a. (5 points) Is the following statement legal in Java? Why?

```
if (expression1) if (expression2) command1;
else command2;
```

b. (5 points) Give the condition under which `command2` will execute. Why?

(4) Grammar (15 points)

a. (9 points) Give the BNF rules of an arithmetic expression. The expression may contain identifiers (variables) and literals (constants), operators (+, -, *, and /) and separators (left and right parentheses).

b. (3 points) Show that the expression (x+y)*2 is syntactically correct

c. (3 points) Show that the expression (x+y)*2) is not.