

Fall 2017 Qualifying Exam – Languages

Your Test ID Number: _____

Instructions

Write your test id number above and on each page of your answers. Read the problems carefully and write answers to all of them. This exam is closed book and closed notes.

Part 1: CS 440 [50 points]

(1) (30 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 exception type checking and no exception type checking, an advantage of exception type checking, and an advantage no exception type checking. We expect your answers to be concise. Points will be deducted otherwise.

- a. (6 points) Exception type checking as in Java vs. no exception type checking as in C#.
- b. (6 points) Pointers in C vs. references in Java.
- c. (6 points) Functional language vs. imperative language.
- d. (6 points) LR(1) vs. LL(1) languages
- e. (6 points) von Neumann vs. lambda calculus

(2) (10 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`.

Qualifying Exam – Languages

```
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 is thrown and handled, 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`.

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

- (3) (10 points) The following set of BNF rules come 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) The following statement is legal with respect to the above rules.

Briefly explain why.

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

- b. (5 points) Will command2 gets executed when expression1 is false or when expression2 is false. Briefly explain.

Part 2: CS 536[50 points]

For questions 1 – 3, show your work and use syntactic or logical transformations as needed to simplify your answer.

- (1) (4 points) Calculate $wlp(\text{if } a < x \square x := x + b \sqcap b < y \square y := y - b \text{ fi}, 0 < x < y)$.
(2) (8 points) Calculate $wp(b[z] := b[w]; b[x] := b[y], b[z] < b[x])$.

Qualifying Exam – Languages

- (3) (8 points) Calculate $sp(x \geq y, y := y/x; y := y/z)$.
- (4) (14 points) Add a final post condition to the program below (make it as strong as you can). Give a full outline for total correctness including invariant and bound function.

```
{true}
X:=1; y:=1; m:=n
While m>0 do
    X:=x+y; y:=y+y;
    m:=m+1
od
```

- (5) (8 points) What are the interference freedom checks for {p1} $x := e$; {q1} $y := f\{r1\}$ and {p2} while B do {q2} $z := g\{r2\}$ od {s2}
- (6) (8 points) List all the deadlock-freedom checks for the following parallel program outline.
[{p1} S1; {q1} **await** B1 do {r1} T1 {s1} U1; {t1}
|| {p2} **await** B2 **do** S2 **end**; {q2} **await** C2 **do** {r2} T2; {s2} U2 **end** {t2}].