Programming Language Ph.D. Qualification Exam

Fall 2020

1. General PL Questions (20pts)

1. Attribute grammars are used to specify
    a. basic syntax of a programming language.
    b. non-finite state machines.
    c. static semantics of a programming language; (d) dynamic semantics of a programming language.
    Answer C

2. Assuming that y is of type String, what happens in the Java statement ``String x = y''?
    a. The wrapper method for a String is called to create a new string object and the result bound to x.
    b. The address of y is modified to be the address of x.
    c. The object bound to y is copied and bound to x, and any previous binding of x to an object is lost.
    d. x and y become aliases.
    e. The contents of y are copied into the storage allocated to x.
    Answer D

3. Which of the following is true of Extended BNF (EBNF) notations:
    a. EBNF notations allow one to describe the semantics of a programming language.
    b. any grammar in a EBNF notation can be also be expressed in a regular BNF notation.
    c. EBNF notations allow one to describe some grammars that can not be described by regular BNF notations.
    d. EBNF can encode attribute grammars where as regular BNF notations cannot.
    Answer B

4. What best distinguishes a purely functional programming language from an imperative one?
    a. There are no variables and hence no assignment operation in a purely functional language.
    b. A purely functional language lacks the ``go to'' statement, but an imperative language always has such a command.
    c. All subprograms must be declared with the keyword function in a purely functional language.
    d. There is no real difference, only a difference in the recommended coding style.
    Answer A

5. The main difference between a sentence and a sentential form is
    a. There is no difference.
    b. A sentence contains only terminal symbols, but a sentential form can contain some non-terminal symbols.
    c. Sentential forms are a subset of sentences, but the converse is not true.
    d. Sentential forms have no handles, but a sentence does.

Answer B

6. The language that is generally regarded as the first object-oriented language is
    a. Java;
    b. C++;
    c. Simula;
    d. Lisp;
    e. COBJOL;
    f. Ada;
    g. PL/I.

    Answer C

7. The Unix YACC utility program parses input using a
    a. recursive descent parser.
    b. a top-down parse.
    c. an RR(1) parser.
    d. a bottom-up shift reduce parser.

    Answer D

8. Axiomatic semantics is often used for
    a. program verification.
    b. byte code generation.
    c. syntax directed parsing.
    d. encoding an unambiguous language specification.

    Answer A

9. Operator associativity generally applies to
    a. only prefix operators.
    b. only infix operators.
    c. only postfix operators.
    d. prefix, infix and postfix operators.

    Answer B

10. A tail-recursive algorithm is generally better than a non-tail recursive algorithm because
    a. it can be run without growing the stack;
    b. it is easier to understand.
    c. it has no side-effects.
    d. all of the above.
    e. none of the above
        i. Answer A

2) GRAMMARS (20pts)

a. Briefly describe what it means for a grammar to be ambiguous (5pts)

Answer: An ambiguous grammar is one in which there is at least one sentence in the language the grammar defines that can be assigned at least two parse trees. Another way to define it is that there is a sentence that has at least one distinct left-most derivation different from a right-most derivation.

b. How can one prove a grammar is ambiguous? (5pts)

Answer: You prove that a grammar is ambiguous by finding a sentence and demonstrating that it has two parse trees or two leftmost derivations.

c. Give an example of an ambiguous grammar and demonstrate that it is ambiguous. (10pts)

Answer: Grammars that have repetition in the RHS are usually ambiguous E -> E + E | id is a simple one. Then show that 3+4+5 has two distinct derivation

3) Find the Weakest Precondition for the following code segment (10pts)

```
x = y;
if (x > 5)
    x = x + y;
else
   x = x * x;
```

{ 0 <= x && x <= 100 }

answer:

Positiive (x > 5)	Negative (x <= 5)
	{ x > -10 } ^ {x < =5}
	{ x < 10 && x > -10} ^ {x<=5}
{ 0 <= x + y && x + y <=100}	{ 0 <= x*x && x*x < 100}
x = x + y;	x = x * x;
{ 0 <= x && x <= 100 }	***{ 0 <= x && x <= 100 }***

Combine: { 0 <= x + y && x + y <=100 && x > 5 } || { x > -10 && x < =5}
substitute x=y
: { 0 <= y + y && y + y <=100 && y > 5 } || { y > -10 && y <= 5}
Algebra:
: { 0 <= 2y && 2y <=100 && y > 5 } || { y > -10 && y < =5} =>
: { 0 <= y && y <=50 && y > 5 } || { y > -10 && y <= 5} =>
: { y <=50 && y > 5 } || { y > -10 && y <= 5} =>
{ -10 < y <= 50}

4) Dynamic Memory Management. (20pts)

a. Give an example which creates garbage in any language. Identify the language.

Answer:
A = new String("hello");
A = new String("bad); // first object is not unreferenced

b. Give an example which creates a dangling pointer/reference

Answer
A = malloc(10)
B = A;
free(A);
// B now points to dynamic memory that has been freed

c. What is the relationship between garbage and dangling pointers?

Answer:

Both concepts deal with dynamic (heap) memory. A dangling is caused when two or more references exists for the same location (alias) and the system allows for the memory to be reused causing the other references either to be invalidated or point to replaced information. Garbage occurs when dynamic is no longer referenced and may be recovered

d. Describe reference counters as they relate to dynamic memory management. Why are the used? How are they used? Are there any limitations

Answer:

Reference counters can be used to aid in eliminating dangling pointers. A reference counter is additional space which indicates how many references exist to that memory location. This technique requires additional space in the heap, time to maintain the reference count and possibly the limitation of the max number of references allowed due to size of reference counter.

5) Operational semantics (20pts)

Write the Operational semantics (using jumps, conditional jumps and labels) of CONTINUE used in FOR loops.

```
for (s1; s2; s3 )
{     s4;
      continue;
      s5;
}
```

Where s1..s5 are statements and expressions as generic placeholders for your solution We are looking for the flow of control as it relates to the continue statement.

Answer:

```
      S1
L1:  S2
      if (S2 is false) jump to L3
     S4
     JUMP L2 // represents continue
     S5
L2:  S3
     JUMP L1:
L3: NOP
```

6) Functional Language (10pts)

Consider the following Scheme function **what**:

```
(define (what x y z)
        (cond ((null? X) 0)
              ((equal? (car x) z) 0)
              ((equal?) (car x) y)
                (+ 1 (what (cdr x) y z)))
              (else (what (cdr x) y z)
```

Briefly describe what this function does:

What is the output of (what '(a b a c b a c) 'a 'c)

Answer:

The function counts the number of occurrences of **y** in **x** up to the first occurrent of **z**.

The result of this function is 2.