**Programming Languages**
**CSCI-GA.2110.001 Spring 2017**

**Homework 1**
**Due Monday, March 13**

You should write the answers using Word, latex, etc., and upload them as a PDF document. No implementation is required. Since there are drawings, if you prefer, you can hand write the answers and scan them to a PDF.

1. Provide regular expressions for defining the syntax of the following. You can only use concatenation, $|$, $*$, parentheses, $\epsilon$ (the empty string), and expressions of the form [A-Z], [a-z0-9], etc., to create regular expressions. For example, you cannot use + or any kind of count variable.

   (a) Passwords consisting of letters and digits that contain at least one upper case letter and one digit. They can be of any length (obviously at least two characters).

   (b) Floating point literals that specify an exponent, such as the following: 243.876E11 (representing $243.867 \times 10^{11}$).

   (c) Procedure names that: must start with a letter; may contain letters, digits, and _ (underscore); and must be no more than 7 characters.

2. (a) Provide a simple context-free grammar for the language in which the following program is written. You can assume that the syntax of names and numbers are already defined using regular expressions (i.e. you don't have to define the syntax for names and numbers).

```
program one;

  var x;

  function f(var x, var y)
    var z;
  begin
    z := x+y-1;
    z := z * 2.0;
    return z;
  end f;

  procedure g()
    var a;
  begin
    a := 3;
    x := a;
  end g;

begin
 g();
 print(f(x));
end one;
```

You only have to create grammar rules that are sufficient to parse the above program.

(b) Draw the parse tree for the above program.

3. (a) Define the terms *static scoping* and *dynamic scoping*.

   (b) Give a simple example, in any language you like (actual or imaginary), that would illustrate the difference between static and dynamic scoping. That is, write a short piece of code whose result would be different depending on whether static or dynamic scoping was used.

   (c) In a block structured, statically scoped language, what is the rule for resolving variable references (i.e. given the use of a variable, how does one find the declaration of that variable)?

   (d) In a block structured but dynamically scoped language, what would the rule for resolving variable references be?

4. (a) Draw the state of the stack, including all relevant values (e.g. variables, return address, dynamic link, static link, closures), during the execution of procedure D.

```
procedure A;
  procedure B(procedure C)
    procedure  D(x:integer);
    begin
      writeln(x);
    end;
 begin
   C(D);
 end;
 procedure F;
   procedure G(procedure H);
    begin
       H(6);
    end;
  begin
      B(G);
  end;
begin
 F;
end;
```

   (b) Explain what the two parts of a closure (as shown in your diagram) are for.

   (c) If both the dynamic link and the return address in a strack frame point back to the calling procedure, what is the difference between them?

5. For each of these parameter passing mechanims,

   (a) pass by value

   (b) pass by reference

   (c) pass by value-result

   (d) pass by name

state what the following program (in some Pascal-like language) would print if that parameter passing mechanism was used:

```
program foo;
  var i,j: integer;
      a: array[1..5] of integer;

  procedure f(x,y:integer)
  begin
    x := x * 2;
    i := i + 1;
    y := a[i] + 1;
  end

begin
  for j := 1 to 5 do a[j] = 10*j;
  i := 1;
  f(i,a[i]);
  for j := 1 to 5 do print(a[j]);
end.
```

6. (a) In Ada, write a procedure that declares two tasks, task one and task two. When the procedure is called, task one and task two should print "one" and "two", respectively, over and over such that the printing of "one" and "two" is perfectly interleaved. Thus, the output should look like:

```
one
two
one
two
...
```

   (b) Looking at the code you wrote for part (a), are the printing of "one" and the printing of "two" occurring concurrently? Justify your answer by describing what concurrency is and why these two events do or do not occur concurrently.