Programming Languages
CSCI-GA.2110.001 Fall 2021

Homework 1
Due Sunday, October 24 at 11:55pm

You should write the answers using Word, latex, etc., and upload them as a PDF document. No implementation is required. Since there are drawings, you can take photos or scan the drawings and upload them separately.

1. Provide regular expressions for defining the syntax of the following. You can only use concatenation, |, $*$, ϵ (the empty string), parentheses, and expressions of the form [A-Z], [0-3], [A-Z0-3], etc., to create regular expressions. That is, you cannot use + or any kind of count indicator.

   (a) Strings consisting of any number of lower case letters, upper case letters, and digits, such that the occurrences of upper case letters and digits are alternating. That is, there cannot be two upper case letters in the string without a digit somewhere in between them and there cannot be two digits in the string without an upper case letter somewhere between them. For example, "aAbc3dEf4ghIjk5m" would be an example of such a string. However, "aAbc3dEfGhi4jk" would not be a valid string, since there is no digit between E and G.

   (b) Positive and negative floating point literals that specify a positive or negative exponent, such as the following: 243.876E11 (representing 243.867×10^{11}) and -135.24E-4 (representing -135.24×10^{-4}). There must be at least one digit before the decimal point and one digit after the decimal point (before the "E").

   (c) Variable names that (1) must start with a letter, (2) may contain any number of letters, digits, and _ (underscores), and (3) may not contain two consecutive underscores.

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). In this language, a program consists of a sequence of function definitions.

```
fun fac 0 = 1
 |  fac n = n * fac(n-1)

fun foo x y =
   let val z = x + y
       fun bar n = n * 2
   in bar z
   end
```

   You only have to create grammar rules that are sufficient to parse the above program. Your starting non-terminal should be called PROG (for "program") and the above program should be able to be derived from PROG.

   As a hint to get you started, here are the first few productions in the CFG that I wrote:
   PROG $\rightarrow$ FUNS

```
FUNS → FUN | FUN FUNS
FUN → fun DEFS
DEFS → DEF | DEF "|" DEFS
```

Feel free to use the above lines, but you certainly don't have to.

(b) Draw the parse tree for the above program, based on a derivation using your grammar. The parse tree will be quite large, so you should separate it across several pages (a subtree on each page).

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) Why do all modern programming languages use static scoping? That is, what is the advantage of static scoping over dynamic scoping?

4. (a) Draw the call stack for the following program that would exist when the print statement is executed. Assume the language is statically scoped. Show at least the local variables and parameters (including any closure), the static and dynamic links, and identify which stack frame is for which procedure.

```
procedure A()
   x: integer := 17;

   procedure B(procedure D)
      x: integer := 20;

      procedure C()
        procedure F(y:integer)
        begin (* F *)
           D(y)
        end; (* F *)
      begin (* C *)
         F(x);
      end; (* C *)

   begin (* B *)
      C();
   end (* B *)

   procedure  E(z:integer)
   begin (* E *)
      print(z, x);
   end; (* E *)

begin (* A *)
  B(E);
```

```
end; (* A *)
```

(b) What would the program print?

(c) Suppose, instead, that the language was dynamically scoped. What would the program print?

5. For each of these parameter passing mechanisms,

   (a) pass by value

   (b) pass by reference

   (c) pass by value-result (assume the address of an argument is computed only once)

   (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..10] of integer;

  procedure f(x,y:integer)
  begin
    x := x * 4;
    i := i + 1;
    y := a[i] * 5;
  end

begin
  for j := 1 to 10 do a[j] = j;
  i := 2;
  f(i,a[i]);
  for j := 1 to 10 do print(a[j]);
end.
```

6. In Ada, define a procedure containing two tasks such that (1) the first task prints the odd numbers from 1 to 99 in order, (2) the second task prints the even numbers from 2 to 100 in order, and (3) for each odd number N, the value of N and N+1 are printed concurrently but there is no other concurrent printing of numbers. For example, 1 and 2 are printed concurrently, and 3 and 4 are printed concurrently, but 1 and 2 must be printed before 3 and 4.