HW3

25pts, no extra credit

Posted Tuesday, October 4, 2022

Due Friday, October 14, 2022

Problem 1 (10pts). Consider the following pseudocode, assuming nested subroutines and static scoping:

```
procedure main
     g : integer

     procedure B(a : integer)
          x : integer

          procedure A(n : integer)
               g := n

          procedure R(m : integer)
               write_integer(x)
               x /:= 2 -- integer division
               if x > 1
                   R(m + 1)
               else
                   A(m)

          -- body of B
          x := a * a
          R(1)

     -- body of main
     B(3)
     write_integer(g)
```

a) (3pts) What does this program print?

b) (3pts) Show the frames on the stack when A has just been called. For each frame, show the static and dynamic links.

c) (4pts) Explain how `A` finds `g`.

Problem 2 (15pts). The expression grammar below generates arithmetic expressions in prefix form:

$$\begin{aligned} E &\rightarrow O\ E\ E \mid \texttt{-}E \mid \texttt{id} \\ O &\rightarrow \texttt{*} \mid \texttt{+} \end{aligned}$$

a) (5pts) Write an attribute grammar to translate expressions into fully parenthesized infix form. For example, expression `* * A + B C D` turns into the following fully parenthesized expression `((A * (B + C)) * D)`.

b) (10pts) Now write an attribute grammar to translate the expressions into *parenthesized* expressions in infix form *without redundant parentheses* assuming the standard convention: unary `-` has highest precedence, followed by `*`, followed by `+`, and `*` and `+` are left-associative. For example, the above expression turns into `A * (B + C) * D`. *Hint:* Assign a precedence attribute *prec* to operators and expressions.