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 grammar below generates Boolean expressions in prefix notation:

```
B  \rightarrow  O B B | \text{not} B | \text{id}
O  \rightarrow  \text{and} | \text{or}
```

a) (5pts) Write an attribute grammar to translate Boolean expressions into fully parenthesized infix form. For example, expression `and and a or b c d` turns into the following fully parenthesized expression `((a and (b or c)) and d).

b) (10pts) Now write an attribute grammar to translate the Boolean expressions into *parenthesized* expressions in infix form *without redundant parentheses*. Use the established convention that `not` has highest precedence, followed by `and`, followed by `or`, and `and and or` are left-associative. For example, the above expression turns into `a and (b or c) and d.`