1) Consider the Pascal program to the right:
   a) What is the reference environment at the location in the program indicated by 
      \textless{}\textequal{} (*)? That is, give the variables, arguments, and procedures that are 
      visible (in scope) at this location.
   b) The main program calls, P1, P1 calls 
      P3, and P3 calls P2. Draw the stack 
      layout of the subroutine stack after 
      these calls. Show the subroutine 
      frames (without their details) with 
      the static links.
   c) Draw the specific subroutine frame 
      layout of procedure P1, indicating the 
      relevant information that it has to 
      contain.

2) Consider the following outline of a program in a C-like language:

   ```
   int add (int i) { return i + d; }
   void p () { const int d = 1;
               print(add(20)); // (1)
   void q () { const int d = 2;
               print(add(20)); // (2)
   ```

   a) If the language is dynamically scoped, what would be printed at points (1) and 
      (2)?
   b) If the language is statically scoped, what would happen?

3) Show a code fragment in which short-circuit semantics for \texttt{or} yield a different result 
   than complete-evaluation semantics.

4) For each of the following mechanisms of the C programming language, show an 
   example of a type error that can be caused by it:
   a) explicit deallocation of memory,
   b) union types, and
   c) pointer arithmetics
5) Show what does the program to the right prints if the programming language has:

a) static scoping and deep binding
b) dynamic scoping and deep binding
c) static scoping and shallow binding
d) dynamic scoping and shallow binding

In which of the four cases above the functional parameter has to be passed as a closure?

```c
int y = 2;
int function f(int function h(int)){
    int y = 3;
    return h();
}
int function g(){
    int x = y+1;
    return x;
}
int function k(){
    int y = 4;
    return f(g);
}
write(k());
```

6) Describe three different ways of allocating in memory a 2-dimensional array A of dimensions N × M.

a) Assuming that indexes range in {0, ..., N-1} and {0, ..., M-1} respectively, give the formula for accessing an arbitrary element A[i][j] for each of the three proposed allocation schemes.

b) Translate the formulas into three-address code

7) Innermost and outermost evaluation strategies may require a different number of steps to evaluate an expression. Show how many steps are necessary to evaluate the expression `square ((1 + 2) * 3)` using the rule `square (x) -> x * x` and the obvious rules for addition and multiplication, by using:

a) innermost (applicative) evaluation
b) outermost (normal order) evaluation
c) outermost evaluation with memoization

8) The ML function `main` to the right computes Fibonacci numbers in a nonstandard way (see Exercise 7.3.1 page 451 of the Dragon book).

Show the stack of activation records that result from a call to `main`, up until the time that the first call (to `fib0(1)`) is about to return. Show the access link in each of the activation records on the stack.

```c
fun main () {
    let
        fun fib0(n) =
            let
                fun fib1(n) =
                    let
                        fun fib2(n) = fib1(n-1) + fib1(n-2)
                        in
                            if n >= 4 then fib2(n)
                            else fib0(n-1) + fib0(n-2)
                        end
                    in
                        if n >= 2 then fib1(n)
                        else 1
                    end
                in
                    fib0(4)
                end;
```