Principles of Programming Languages [PLP-2014]

Exercises on Programming Languages - May 22, 2015

1) Consider the Pascal program to the right:
   a) What is the reference environment at the location in the program indicated by <== (*)? 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.

```
program scopes(input, output)
    procedure P1(A1 : integer)
        var X : integer
        procedure P2(A2 : integer)
            var Y : integer
        begin (* body of P2 *)
            ... <== (*)
        end;
        procedure P3(A3 : integer)
            var X : integer;
        begin (* body of P3 *)
            P2(X)
        end
    begin (* body of P1 *)
        P3(X)
    end
begin (* body of main program *)
    P1(0)
end.
```

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) In the program to the right, for each of the parameter passing modes listed below show the value printed by the program (the same parameter passing mode is applied to both arguments of addto):

- by value
- by reference
- by value/result

```
var z : integer; /* global variable */
procedure addto(x, y)
    begin
        z := 1;
        y := y + x
    end
begin /* body of main program */
    z := 2;
    addto(z, z);
    write_integer(z)
end
```

4) Show a code fragment in which short-circuit semantics for **or** yield a different result than complete-evaluation semantics.

5) FORTRAN only passes parameters in reference mode. C only passes parameters in value mode. Pascal allows both modes. Show how you can get the effect of reference mode in C and how you can get the effect of value mode in FORTRAN by appropriate programming techniques. In particular, show in both FORTRAN and C how to get the effect of the following code.

```
1 variable X, Y : integer;
2 procedure Accept(A:reference integer; B:value integer);
3    begin
4         A := B;
5         B := B+1;
6    end; -- Accept
7 X:=1;
8 Y:=2;
9 Accept(X, Y);
10 -- at this point, X should be 2, and Y should be 2
11
```

6) C does not allow a procedure to be declared inside another procedure, but Pascal does allow nested procedure declarations. What effect does this choice have on runtime storage organization?

7) Define, in any programming language, a function, f , such that the evaluation of the expression (a + f (b)) ∗ (c + f (b)) when performed from left-to-right has a result that differs from that obtained by evaluating right-to-left.

8) Consider the following function:

```
int foo (int x){
     if (x>100)  return x-10;
     else  return foo(foo(x+11));
}
```

Is this tail recursive? Justify your answer.

9) The following code fragment is written in a pseudo-language which admits bounded iteration (numerically controlled) expressed using the **for** construct.

```
z=1;
for i=1 to 5+z by 1 do{
     write(i);
     z++; }
write(z);
```

What is printed by `write`?

10) Show what does the following program 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

```
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());
```

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

11) The following code is in a language with static scope and call by name, where each binary operation is evaluated left-to-right. What does it print?

```
{int x=5;
 int P(name int m){
       int x=2;
       return m+x;
}
 write(P(x++) + x);
}
```

12) Write in your preferred language an iterator (or a *true* iterator) acting on lists, that returns its element in reverse order. Discuss the design choices.

13) Why the runtime stack can be eliminated by translating programs to Continuation Passing Style?

14) Transform the following simple function definition in *continutation passing style*

```
function volume (int x, int y, int z){
      return x * y * z;
}
```

15) Write a simple program where *call by name* and *call by need* produce different results.