1 Short Answer Questions

1. (1 point) Pcode is an example of intermediate code. True or false? **Answer:** True—see p.48.

2. (1 point) Suppose that an experienced programmer is using a programming environment that provides both an interpreter and a compiler for a programming language. The programmer will use the compiler while developing a program and the interpreter to generate efficient code after the program has been validated. True or false? **Answer:** False—see p.49: switch "compiler" and "interpreter."

3. (1 point) In a static language, the type a variable is bound to the variable name at program execution ("run") time. True or False? **Answer:** False—at compile time.

4. (2 points) Why do some languages distinguish between the declaration and the definition of a routine? **Answer:** Because of mutual recursion. (Partial credit for: because it encapsulates the interface of a group of related functions, e.g. for methods of a class.)

5. (2 points) Define overloading. **Answer:** See p.68 text.

6. (2 points) What is aliasing? **Answer:** See p.67 text.

7. (1 point) In Simplexem, the effect of the instruction `set 10, D[20]` is to copy the value stored at location 10 of data memory into location 20 of data memory. True or false? **Answer:** False. The reverse is true.
2 Run-time Structure

(Carlo Ghezzi) Consider the following program fragment.

```plaintext
program pippo{
    integer a,b;
    routine gamma(); // declaration of gamma
    routine alpha(){
        integer a, x, y;
        routine beta(){
            integer z, g;
            a = z + x + b; // (i)
        }
        .......
        x = a + y; // (ii)
        .......
    }
    .......
};
routine gamma(){ // definition of gamma
    integer a, x, z, w;
    .......
};
    .......
};
```

1. (4 points) Draw the static nesting tree for this program. Answer: pippo is root; one child of pippo is gamma, which is a leaf; the other child is alpha, which has beta as child. beta is a leaf.

2. (3 points) Compute the distance attribute for the variables that appear in instruction (i). Answer: a:1, z:0, x:1, b:2.

3. (3 points) Compute the distance attribute for the variables that appear in instruction (ii). Answer: all are zero

4. (10 points) Sketch Simplesem memory, showing static and dynamic links in the activation records, for the following sequence of calls: pippo calls gamma, which calls alpha, which calls beta, which calls gamma, which calls alpha.

3 ML

1. (15 points) Write a function fact that computes the factorial of a non-negative integer. Use patterns. Answer:

   ```plaintext
   - fun fact 0 = 1
   = |    fact i = i * fact (i-1);
   val fact = fn : int -> int
   ```
2. (5 points) What does the following function do?

```haskell
fun power(x,k): real =
    if k=1 then x
    else if k mod 2 = 0 then power(x*x, k div 2)
        else x* power(x*x, k div 2);
```

**Answer:** It computes $x^k$ (in log time)

3. (15 points) The following function inserts a real number in the correct place into an ordered list of real numbers. The list is ordered in ascending order.

```haskell
fun ins (x, []): real list = [x]
    ins (x, y::ys) =
        if x <= y then x::y::ys (* x belongs here *)
        else y::ins(x,ys);
```

Use `ins` to write an insertion sort function. Your function, which you should call `insort`, takes a list of reals and produces an ordered list of reals. Use patterns. **Answer:**

```haskell
fun insort nil = nil
    insort (x::xs) = ins (x, (insort (xs)));
```

(* Example of Use
insort [];
val it = [] : real list
- insort [1.0];
val it = [1.0] : real list
- insort [2.0, 1.0];
val it = [1.0, 2.0] : real list
- insort [1.0, 3.0, 2.5, 4.3, 1.1, 0.9];
val it = [1.0, 2.5, 3.1, 4.3, 1.1, 0.9] : real list
- insort [0.9,1.0,1.1,2.5,3.1,4.3];
```
4. (10 points) Consider the ML session printed below.

```ml
- fun cond(test, then_part, else_part) =  
  if test then then_part else else_part;
= val cond = fn : bool * 'a * 'a -> 'a
- val x = 0;
val x = 0 : int
- if x = 0 then [0] else tl[];
val it = [0] : int list
- cond(x=0, [0], tl[]);
uncaught exception Empty
raised at: boot/list.sml:37.38-37.43
```

Explain briefly why `if` and `cond` behave differently.

**Answer:** `if` uses lazy evaluation, `cond` uses eager (strict) evaluation, which is the default (for user-defined functions) in ML.

5. (10 points) While we did not study higher-order functions in ML, they exist. Here is an example of definition and use of a higher-order function in ML.

```ml
- fun simpleMap(F, nil) = nil
  | simpleMap(F, x::xs) = F(x)::simpleMap(F,xs);
= val simpleMap = fn : ('a -> 'b) * 'a list -> 'b list
- fun plus2 x = x + 2;
val plus2 = fn : int -> int
- plus2 3;
val it = 5 : int
- simpleMap (plus2, [1,2,3,4,7]);

What is it now? **Answer:** `val it = [3,5,6,9] : int list`

What does `simpleMap` do? **Answer:** It applies its first argument to each element of its second argument.