TIE-20306 Principles of Programming Languages *(Matti Rintala)*

Exam 10.3.2014

No literature, calculators, or computers in the exam.

Some advice on answering:

1. You can answer the questions *either in English or Finnish*. If you answer in Finnish, you can still use English terms if you don't remember/know the corresponding Finnish term.
2. Remember to answer to *all* questions, if a question contains several sub-questions.
3. If an answer requires writing program code, absolute syntactic correctness is not required (unless the question is specifically about syntax).

1. Terms. Explain the following terms and their use in programming languages. Give *also* an example, if possible.
   a) Activation record
   b) Static scoping
   c) Ambiguous grammar
   d) Dynamic typing
   e) Logic programming paradigm
   f) Parse tree (Derivation tree)

2. Compilation
   a) What are the typical phases of compilation? Explain what happens in each phase.
   b) In which phase of compilation do the following things belong? (C++ is used as an example language.)
      i. An error when a semicolon is missing between two statements.
      ii. An error when a function call and the corresponding function definition contain a different number of parameters.
      iii. A check that in `a=f()` the return value of the function `f` can be assigned to variable `a`.
      iv. Producing an implicit type conversion if the return value in the previous item cannot be assigned directly.
      v. An error when a string literal is missing the final quotation mark.
      vi. Not evaluating an expression, if its result is multiplied by zero.
      vii. An illegal character `@` is used in a variable name.
      viii. Structure of the activation record of a function is determined.
      ix. An error when a function call is missing the final parenthesis.

.... Turn the page! ....

3. Functional programming

   a) In which ways do functional programming languages typically differ from imperative languages? What are the benefits? Does functional programming bring any disadvantages?

   b) What is lazy evaluation (found in Haskell language, for example)? What are its benefits? What are its disadvantages? Why is it suitable especially for functional programming languages?

   c) Give an example of programming structures, which are possible (or easy) in languages supporting lazy evaluation, but impossible or really difficult without it.

   d) What are lambda functions (also typical in functional programming languages)? How are they useful (give examples)?

4. Parameter passing

   a) Explain what parameter passing mechanisms exist in programming languages. For each mechanism, give its name, explain its principles and tell in what kind of situations the mechanisms is useful.

   b) Below is a short C++ program using a fictional parameter passing mechanism marked “%” (line 8). For *each* different parameter passing mechanism, tell what the program would print out if function f used that mechanism for its parameters.

```
 1 #include <iostream>
 2 using namespace std;
 3
 4 int a[2] = {1, 2};
 5
 6 void printout();
 7
 8 void f(int% x, int% y)
 9 {
10   x = 1;
11   cout << "x=" << x << endl;
12   y = y + 1;
13   cout << "y=" << y << endl;
14   printout();
15 }
16
17 void printout()
18 {
19   cout << "a[0]=" << a[0] << endl;
20   cout << "a[1]=" << a[1] << endl;
21 }
22
23 int main()
24 {
25   int i = 0;
26   f(i, a[i]);
27   cout << "i=" << i << endl;
28   printout();
29 }
```