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

Exam 9.12.2013

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).

.... **Answers 1. & 2. on a separate sheet of paper!** ....

1. Terms. Briefly explain the following terms and compare them. What are their benefits compared to each other?
   - Stack dynamic variable vs. heap dynamic variable
   - Static typing vs. dynamic typing
   - Static scoping vs. dynamic scoping
   - Pass-by-value vs. pass-by-result
   - Bounded iteration vs. unbounded iteration
2. Phases of compilation. Below is a short C++ code snippet:

```
int f(std::string* p; int* q)
{
  double i = p.length()/*q
  std::cout < i < " is a number';
}
```

   a) What phases does compiling typically contain? What is done is each phase?

   b) Divide the code into lexemes (copy the lines to your answer sheet and separate lexemes with vertical lines).

   c) Does the code have error(s) which would be noticed during lexical analysis? **Tell why** the error(s) are specifically lexical. Fix the error(s).

   d) Pick at least three different syntactic structures from the code and explain them (in your answer, indicate where in the code the structures begin and end).

   e) After fixing the possible lexical errors, does the code contain errors that would be noticed during syntactic analysis? **Tell why** the error(s) are specifically syntactic. Fix the error(s).

   f) After fixing the possible lexical and syntax errors, does the code contain errors that would be noticed during semantic analysis? **Tell why** the error(s) are specifically semantic. Fix the error(s).

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

.... **Answers 3. & 4. on a separate sheet of paper!** ....

3. Paradigms

   - What are the main characteristics of the *functional programming paradigm?*
   - What benefits does functional programming have compared to imperative programming?
   - What is *lazy evaluation* that is often used in functional programming? Why is it suitable especially for functional programming compared to other paradigms?

4. Activation records and subroutines

   - What is an *activation record*, what does it contain, and what is it used for?
   - Based on the code below, draw a picture showing the contents of the execution stack just before that return-statement of function f is executed. The picture should also show variables that are located outside the stack, as well as where pointers (and references) point to.

```
#include <vector>

int f(int a, int& b)
{
  int sum = a + b;

  std::vector<int>* v = new std::vector<int>;
  v->push_back(a);
  v->push_back(b);

  return sum + v->size();
  // A memory leak... Sigh, don't care about it...
}

int main()
{
  int x = 3;
  f(4, x);
}
```