0.2. Defining lamer

We have often been asked what “lamer” means, when referring to a programmer. Again, without giving a formal definition, we will explain the term using an introduction a’ la Karl May. [*May-1893*].

First of all, the most important feature of a lamer is that he does not even suspect to be one.

Further, one of the most widely spread definitions for “lamer” is “a programmer, who writes programs in column and who, on hearing of “recursion” gets a slight diarrhea.

“Lamer” is a programmer, who claims he knows several programming languages, but who in fact does not even have an idea of algorithms.

“Lamer” is a programmer, who when using comments in his coding, if at all, makes them of the kind:

```
i++;       /* incrementing i by 1*/
```

“Lamer” is a programmer, who does not know that behind the simple game ‘Minesweeper’ for Windows stays one of the most serious problems in informatics, for the solving of which Stanford University has allotted a prize of one million dollar. [*Prize-2000*].

“Lamer” is a programmer, who can write *quick* sorting only if he learns it by heart. In fact, the true programmers can rarely memorize it and write it quickly. They will rather invent it on their own for some 10 minutes.

“Lamer” is a programmer, who does not know that “the garbage collector” has not been invented to collect the corpses of the ‘dead’ objects in Java. Rather, behind it stays a nice and powerful theory, presented as early as in the middle of the 20-th century.

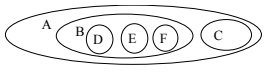
“Lamer” is a programmer, who does not know that the most effective searching engine in Internet - Google, gives such relevant results not only because of the processing “iron”, but because it is based on innovative methods from the graph theory.

“Lamer” is a programmer, who does not know that the formula for the effective accomplishment of any computer project is: some specifications + many and effective tehcniques.

“Lamer” is a programmer, who cannot comprehend why a program, for which someone has said to be of complexity $\Theta(n^2)$ is faster than one of complexity $\Theta(n^3)$, when $n^2 > n^3$.

“Lamer” is a programmer, who can program in C and who can find the least common denominator of two numbers on a sheet of paper, but who cannot write a program for that.

“Lamer” is a programmer, who cannot explain to himself why when compressing a file, and then when compressing it again, its size doesn’t shrink any further.

“Lamer” is a programmer, who knows that this  is a tree, but does not suspect that this  is also a tree, just as this 

is a tree as well. And that, in fact, he himself as a programmer is also a young and green tree.

“Lamer” is a programmer, who for 10 minutes can make given program 10 times slower without even a slight worrying that someone before him has spent 10 hours in optimizing it, only to make it 10% faster.

“Lamer” is a programmer, who claims he ‘brute-forces’ FTP passwords (*FTP*: *File Transfer Protocol* – a protocol for transferring files over the net), without noticing that with the current speed of Internet he could hardly even ‘hack’ a password longer than 5 symbols (for which he would need just a simple combinatory computation), unless he’s got crazy luck.

“Lamer” is a programmer, who could hardly comprehend an association of a nice dinner with a nice program (the quality of which both depends on the used recipe/algorithm).

“Lamer” is a programmer, who does not know that the shortest distance between two points is not always the piece of straight line that connects them (sometimes it is being found by the Dijkstra algorithm).

“Lamer” is a programmer, for whom ‘binary search’ means ‘searching in binary files’.

“Lamer” is a programmer, who awes at programming fragments like:

```
(0x000000FF & (i >> 24);
```

often only because he does not have an idea what exactly they mean, and who, seized by panic, goes forward to the next page on seeing a hieroglyph like:

$$\chi^2 = \frac{n}{m}\sum_{i=1}^{n}\left(f_i - \frac{m}{n}\right)^2$$

“Lamer” is a programmer, who has not heard of ‘dynamic programming’. Actually, one of the really good ways to defeat a lamer in an argument is to tell him: “Gee, this is a well-known problem and it can be solved easily, elegantly and effectively through dynamic programming!”

Initially, like all freshmen, we were also nothing more than two young and green “lamers”. In the course of time, armed with good books and articles, and most of all by hard working on solving algorithmic problems, we started to grow over that level. Today we are ready to help you, too, dear reader, to run away from the sterility of the ‘manufactured programming’.

We will end this paragraph with one beautiful thought of another famous writer (Mark Twain):

“*When I want to read something nice, I sit down and write it myself.*”

The aim of this book is to bring you, dear reader, to the state, at which you will be able to confidently state:

“*When I want to see a nice piece of coding, I sit down and write it myself*.”