The original instructions sent to participants in the study described in Lisp as an Alternative to Java

Greetings fellow Lispers!

Thank you for agreeing to participate in this study, which is a followup to a recently published study comparing C++ and Java across several metrics. The task description below is a lightly edited version of the one that was used in that study. (Note: the original study was done in Germany, and not by me. Please don'y try to look up the original until you are done coding.)

The purpose of this followup is to find out if Lisp is competitive with C++ and Java in terms of the metrics used in the original study. These metrics include run time, memory usage, and development time. In order to measure development time I am going to ask that you try to keep track of how much time you actually spend woking on this task. (Don't stress out over it too much; this is an informal study.)

When you are ready to begin, read the task description below and have at it. When you are done just email your code to me, along with your estimate of how long you spent on it. I will accept entries for one week (i.e. until midnight next Monday). This is not a super-hard deadline, but I do want to keep this from dragging on indefinitely.

Thank you again for volunteering your time.

Erann Gat
[email address elided]

Task description

Attention: Please follow these instructions super accurately.

First read through the whole of it in order to get an overview. Concentrate on the details only upon second reading.

The following mapping from letters to digits is given:

E	J N Q	R W X	D S Y	F T	A M	C I V	B K U	L O P	G H Z
e	j n q	r w x	d s y	f t	a m	c i v	b k u	l o p	g h z
0	1	2	3	4	5	6	7	8	9

We want to use this mapping for encoding telephone numbers by words, so that it becomes easier to remember the numbers.

Functional requirements

Your task is writing a program that finds, for a given phone number, all possible encodings by words, and prints them. A phone number is an arbitrary(!) string of dashes - , slashes / and digits. The dashes and slashes will not be encoded. The words are taken from a dictionary which is given as an alphabetically sorted ASCII file (one word per line).

[NOTE: The dictionary is in German and contains umlaut characters encoded as double-quotes. The double-quotes should be ignored. EG.]

Only exactly each encoding that is possible from this dictionary and that matches the phone number exactly shall be printed. Thus, possibly nothing is printed at all. The words in the dictionary contain letters (capital or small, but the difference is ignored in the sorting), dashes - and double quotes " . For the encoding only the letters are used, but the words must be printed in exactly the form given in the dictionary. Leading non-letters do not occur in the dictionary.

Encodings of phone numbers can consist of a single word or of multiple words separated by spaces. The encodings are built word by word from left to right. If and only if at a particular point no word at all from the dictionary can be inserted, a single digit from the phone number can be copied to the encoding instead. Two subsequent digits are never allowed, though. To put it differently: In a partial encoding that currently covers k digits, digit k+1 is encoded by itself if and only if, first, digit k was not encoded by a digit and, second, there is no word in the dictionary that can be used in the encoding starting at digit k+1.

Your program must work on a series of phone numbers; for each encoding that it finds, it must print the phone number followed by a colon, a single(!) space, and the encoding on one line; trailing spaces are not allowed. All remaining ambiguities in this specification will be resolved by the following example. (Still remaining ambiguities are intended degrees of freedom.)

Sample dictionary:

```
an
blau
Bo"
Boot
bo"s
da
Fee
fern
Fest
fort
je
jemand
mir
Mix
Mixer
Name
neu
o"d
Ort
so
Tor
Torf
Wasser
```

Sample phone number list:

```
112
5624-82
4824
0721/608-4067
10/783--5
1078-913-5
381482
04824
```

Corresponding correct program output (on screen):

```
5624-82: mir Tor
5624-82: Mix Tor
4824: Torf
4824: fort
4824: Tor 4
10/783--5: neu o"d 5
10/783--5: je bo"s 5
10/783--5: je Bo" da
381482: so 1 Tor
04824: 0 Torf
04824: 0 fort
04824: 0 Tor 4
```

Any other output would be wrong (except for different ordering of the lines).

Wrong outputs for the above example would be e.g.

562482: Mix Tor, because the formatting of the phone number is incorrect,

10/783--5: je bos 5, because the formatting of the second word is incorrect,

4824: 4 Ort, because in place of the first digit the words Torf, fort, Tor could be used,

1078-913-5: je Bo" 9 1 da , since there are two subsequent digits in the encoding,

04824: 0 Tor , because the encoding does not cover the whole phone number, and

5624-82: mir Torf , because the encoding is longer than the phone number.

Quantitative requirements

Length of the individual words in the dictionary: 50 characters maximum.
Number of words in the dictionary: 75000 maximum
Length of the phone numbers: 50 characters maximum.
Number of entries in the phone number file: unlimited.

Quality requirements

Work as carefully as you would as a professional software engineer and deliver a correspondingly high grade program. Specifically, thoroughly comment your source code (design ideas etc.).

The focus during program construction shall be on correctness. Generate exactly the right output format right from the start. Do not generate additional output. We will automatically test your program with hundreds of thousands of phone numbers and it should not make a single mistake, if possible -- in particular it must not crash. Take youself as much time as is required to ensure correctness.

Your program must be run time efficient in so far that it analyzes only a very small fraction of all dictionary entries in each word appending step. It should also be memory efficient in that it does not use 75000 times 50 bytes for storing the dictionary if that contains many much shorter words. The dictionary must be read into main memory entirely, but you must not do the same with the phone number file, as that may be arbitrarily large.

Your program need not be robust against incorrect formats of the dictionary file or the phone number file.

Please use only a single source program file, not several source modules. And please work alone.

A large dictionary, which you can use for testing purposes, is available at [original URL elided]