Assignment 1 - Software I, Spring 2004 (0368-2157-09/12)

`http://www.cs.tau.ac.il/~efif/courses/Software1_Spring_04`

Due: March 30, 2004

Required Knowledge getchar, putchar, printf, scanf, while, for, numeric types (int, long, double, float, etc.), basic `UNIX`, some math.h functions (pow, sqrt, log10, etc.).

Before starting to answer the questions, please read very carefully the "Submission Guidelines"[1]. Make sure your programs detect invalid input data, and print out appropriate error messages. Do not add "friendly" messages to your programs, as they are tested automatically by other programs.

Ex 1.1 get_digit

Write a program that reads a decimal number x (positive or negative), prints the number of digits of x, reads an index i of a digit (index 1 means least significant digit of x) and prints the digit of x at index i. If i is not a valid index of digit, the program prints an error message and terminates. Use `long` type to represent x; use an appropriate format in `scanf` and `printf`. To perform the computations, you are not allowed to use loops. Instead, use `log10`, `pow` and `labs` functions from the standard math library. To use these functions, add the line

```
#include <math.h>
```

at the top of the source file, and link with the math library (add `-lm` to the command line) to use these functions.

Ex 1.2 get_digit2

Write the program from Ex. 1.1 without using functions from the library `math.h`.

Ex 1.3 dec2eng

Write a program that converts the decimal representation of a non-negative integer to the base-27 representation. The digits used in the base-27 representation are the space ' ' and the English letters '`a`' through '`z`'. The space represents 0, '`a`' represents 1, '`b`' represents 2,..., and '`z`' represents 26.

$$l_n, l_{n-1}, ..., l_0 = \sum_{i=0}^{n}(l_i \times 27^i)$$

Each place value to the left is equal to 27 times the place value to the right, which implies that each place value to the right is equal to the place value to the left divided by 27.

[1] `http://www.cs.tau.ac.il/~efif/courses/Software1_Spring_04`

The program dec2eng should scan in a non-negative integer in decimal format (you may assume that, without checking), and print out the number in base-27 format (in English letters). You may assume that the input number is smaller than 2^{32}.
Examples:

$$\begin{aligned}
\texttt{a} &= 1 = 1 \\
\texttt{aaaaaa} &= 1 \times 27^5 + 1 \times 27^4 + 1 \times 27^3 + 1 \times 27^2 + 1 \times 27^1 + 1 \times 27^0 = 14900788 \\
\texttt{abcdef} &= 1 \times 27^5 + 2 \times 27^4 + 3 \times 27^3 + 4 \times 27^2 + 5 \times 27^1 + 6 \times 27^0 = 15473895 \\
\texttt{hello} &= 4359030
\end{aligned}$$

Ex 1.4 eng2dec

The program eng2dec should scan in a sequence of english letters or spaces and print out the decimal number that corresponds to it. If the number is greater than 2^{32}, the program terminates with an error. If the input contains illegal characters, the program terminates with an error. You can test eng2dec and dec2eng programs by running one on the output of the other (mind the direction and the newlines).

Ex 1.5 maxbin2eng

Write a program that reads the number of digits in a binary format, and prints out the largest number that can be represented in this format, but it prints it out in base-27 format (in English letters). Example:

```
maxbin2eng
8
il
```

Ex 1.6 e

Write a program that reads a small real number ϵ. It computes an approximation of the natural base[2] e with precision ϵ, according to the formula:

$$e = \sum_{k=0}^{\infty} \frac{1}{k!}$$

The computation should stop when the absolute value of the last element summed is equal to or smaller than ϵ. Use `double` type to represent any real number in the program, and print out the result with 9 digits of precision for the fraction part. Examples:

```
e
0.001
2.718253968

e
0.00000000042
2.718281828
```

[2] For more information about the natural base e see the website `http://mathworld.wolfram.com/e.html`.

Good Luck!

More Information on the Submission

Files Name

The files for the exercises should be located under `~/software1/assign1`, and their names should match the name of the exercise. For example, the `C` source file for exercise 1.1, namely `get_digit`, should be `~/software1/assign1/get_digit.c`. Note that names are case sensitive (i.e. `ex1.C` is different than `ex1.c`).

Giving Permission to the Files

Make sure that you have correct permissions for all the directories and files. Set permissions by executing the following command:

```
chmod 705 ~ ~/software1 ~/software1/assign1 ~/software1/assign1/*
```

Sample Files

In order to test your programs, we have prepared some input and output files which you may compare to your own output. You may use the command `diff` or `diff -b` in order to compare your files to the sample files.

In order to read the input from a file and to write your output into a file you should use redirection. For example:

```
get_digit < infile > outfile
```

The input and output data files may be found under ~tochna1/software1/assign1/test/. The names of these files match the names of the corresponding programs, where the input files have a `.in` extension, and the output files have a `.out` extension.