Lab 4

COMP9021, Session 1, 2013

The aim of this lab is to:

- practice the use of the `getchar()` and `putchar()` functions;
- implement a few essential techniques to process text and operate on characters.

1 A triangle of characters

Write a program that gets a strictly positive integer *n* as input and outputs a triangle of height *n*, following this kind of interaction:

```
Enter strictly positive number: 13
            A
           BCB
          DEFED
         GHIJIHG
        KLMNONMLK
       PQRSTUTSRQP
      VWXYZABAZYXWV
     CDEFGHIJIHGFEDC
    KLMNOPQRSRQPONMLK
   TUVWXYZABCBAZYXWVUT
  DEFGHIJKLMNMLKJIHGFED
 OPQRSTUVWXYZYXWVUTSRQPO
ABCDEFGHIJKLMLKJIHGFEDCBA
```

To prompt the user for an input and store that input in a variable, say `height`, declared as an `int`, include the `p_prompt.h` header file with the directive

```
#include <p_io.h>
```

and let your code contain

```
p_prompt("Enter strictly positive number: ", "%>=1d", &height);
```

Recall that the input should end in carriage return followed by Control D (Control Z in Windows).

Note the `"%>=1d"` format string: it expresses that we want to input a **d**ecimal number greater than or equal to 1. We will be prompted again and again till we indeed provide an integer at least equal to 1 that can fit in a `int` (of course, our program won't produce any readable output when `N` becomes too large...).

2 ☞ Computing statistics on the characters in a text

Write a program that:

- outputs the number of blank characters (spaces, tabs and new lines)
- outputs the length of the shortest word (any sequence of nonblank characters), and
- outputs the length of the longest word (any sequence of nonblank characters)

for data provided from standard input. Data can either be entered from the keyboard, possibly over many lines, end in carriage return followed by Control D (Control Z in Windows). Alternatively, it could be stored in a file *file* whose content is redirected to standard input—hence the program should be run by typing `a.out <`*file* from the command line or by typing `r <`*file* from within Emacs. Use `getchar()` to read the input. Test your program on files that correspond to limiting cases (*e.g.*, an empty file, a file containing nothing but new lines, etc.).

Here are examples of possible inputs and outputs using the first method.

```
$ a.out

   This is a 1rst example
   with input entered from the keyboard
   over 6 lines (2 of which are blank).
   Note that "(2" and "blank)." above are considered as "words".

Input contains 43 blanks, tabs and new lines
Length of shortest word: 1
Length of longest word: 10

$ a.out
And here is another example with only one line of input!!!!!!!!!
Input contains 11 blanks, tabs and new lines
Length of shortest word: 2
Length of longest word: 14
```

3 ☞ Converting a number from one base to another

Write a program that prompts the user for a nonzero hexadecimal number, uses `getchar()` to read the input, and prints out its value in decimal.

The input can be preceded or followed by blanks and will end in carriage return (anything input after the first carriage return will be ignored). An error message should be output in response to inputs that are incorrect, that is, do not start (after possibly some spaces) with `0x` or `0X` followed by hexadecimal digits, or are too large to fit in an `unsigned long`, or are followed by anything but spaces before carriage return.

Here are examples of possible inputs and outputs on a machine where an `unsigned long` occupies 8 bytes.

```
$ a.out
Enter a nonzero hexadecimal number: 0x0
Input is incorrect
$ a.out
Enter a nonzero hexadecimal number: 12
Input is incorrect
$ a.out
Enter a nonzero hexadecimal number: 0x012
Input is incorrect
$ a.out
Enter a nonzero hexadecimal number: 0x12
The decimal value of 0x12 is 18
$ a.out
Enter a nonzero hexadecimal number: 0x12 0
Incorrect input
$ a.out
Enter a nonzero hexadecimal number: oX2AF
Input is incorrect
$ a.out
Enter a nonzero hexadecimal number: 0X2AF
The decimal value of 0x2af is 687
$ a.out
Enter a nonzero hexadecimal number: 0xaBcDeF
The decimal value of 0xabcdef is 11259375
$ a.out
Enter a nonzero hexadecimal number: 0xffffffffffffffff
The decimal value of 0xffffffffffffffff is 18446744073709551615
$ a.out
Enter a nonzero hexadecimal number: 0x10000000000000000
Input is too large or incorrect
$ a.out
Enter a nonzero hexadecimal number: 0X9018345672809172635489
Input is too large or incorrect
```