Lab 11

COMP9021, Session 1, 2014

The aim of this lab is to practice further some of the techniques used so far and that are likely to be tested in the practical exam.

1 ☞ Extracting words from input

Write a C program that prompts the user to input on the next line a sequence of letters, blanks (simple spaces or tabs) and punctuation marks (commas, full stops, colons, question marks or exclamation marks) and displays the output as follows.

- Words (maximal nonempty sequences of consecutive letters) have their first letter displayed in upper case and the remaining letters in lower case.
- Any punctuation mark that does not follow a word with possibly spaces in-between is ignored.
- Any punctuation mark that follows a word with possibly spaces in-between is displayed next to the word.
- Successive words with possibly a punctation mark attached to their ends are separated by a single space.
- The first word, if any, is not preceded by any space.

Your program should behave as follows:

```
$ a.out
Enter a sequence of letters, blanks and punctuation marks on one line:


$ a.out
Enter a sequence of letters, blanks and punctuation marks on one line:
HI
Hi
$ a.out
Enter a sequence of letters, blanks and punctuation marks on one line:
   hi there  !!!!
Hi There!
$ a.out
Enter a sequence of letters, blanks and punctuation marks on one line:
   ??? How    ARE yOu     tONIght    ???? Well?!!? very...WELL
How Are You Tonight? Well? Very. Well
```

```
$ a.out
Enter a sequence of letters, blanks and punctuation marks on one line:
  ..  ! I am    fEeLinG   ..!  .?  QUITE welL   !. ?. .  very...very!.!WELL
I Am Feeling. Quite Well! Very. Very! Well
```

2 ☞ Finding the longest, lexicographically last word

Write a C program that takes as input letters and spaces, possibly over many lines, with carriage return followed by Control D indicating the end of input, and displays the output as follows.

- If some word (maximal sequence of consecutive letters) in the input contains more than 10 letters then the program outputs an error message.
- Otherwise, if no word has been input then the program outputs another error message.
- Otherwise, the program announces the "winner" and outputs, amongst the words of maximal length, the one that comes last in alphabetical order as given by the ASCII character set (where uppercase letters come before lowercase letters).

Your program should behave as follows (recall that the input ends in a final carriage return followed by Control D):

```
$ a.out
  abc efg
   abcdefghijk
  uvw xyzq
Some word is too long!
$ a.out


No word has been input!
$ a.out
   abcdefghij
The winner is: abcdefghij
$ a.out
   a bc def
 ghi k lm
   adf l
The winner is: ghi
$ a.out

    aaa bbb ccc ddd eee ddd ccc bbb aaa zz
The winner is: eee
$ a.out
```

```
abc Abc zh l
The winner is: abc
```

3 ☞ Classifying words depending on how many uppercase letters they contain

Write a program that prompts the user to enter some text (possibly with spaces—that is, simple spaces or tabs) ending in a new line character and behaves as follows.

- If more than 30 nonspace characters are input before the final new line character then the program prints out

  ```
  Too many nonspace characters!
  ```

  and stops.

- Otherwise, if some nonspace character is neither a lowercase nor an uppercase letter then the program prints out

  ```
  Some incorrect characters!
  ```

  and stops.

- Otherwise, if some word in the input (that is, maximal sequence of letters) consists of more than 6 letters then the program prints out

  ```
  Some word is too long!
  ```

  and stops.

- Otherwise, if only spaces have been input then the program prints out

  ```
  No word has been input!
  ```

  and stops.

- Otherwise, for all $l \in \{0, 1, 2, 3, 4, 5, 6\}$, if the input contain some word with exactly l occurrences of uppercase letters then the program prints out:

  Words with l upper case letters:

  followed by all words with l uppercase letters, output in the order in which they have been input.

Here are sample outputs of how your program should behave:

```
$ a.out
Enter words:     this  is a sequence with     31  characters
Too many nonspace characters!
$ a.out
Enter words: But    this sequence    has 30    characters
Some incorrect characters!
$ a.out
Enter words: Too many letters in some word
Some word is too long!
$ a.out
Enter words:
No word has been input!
$ a.out
Enter words: Here is a NUMbER OF WorDs
Words with 0 upper case letters:
   is
   a
Words with 1 upper case letters:
   Here
Words with 2 upper case letters:
   OF
   WorDs
Words with 5 upper case letters:
   NUMbER
$ a.out
Enter words: AND heRe is a LONGER WORD
Words with 0 upper case letters:
   is
   a
Words with 1 upper case letters:
   heRe
Words with 3 upper case letters:
   AND
Words with 4 upper case letters:
   WORD
Words with 6 upper case letters:
   LONGER
```

4 ☞ Finding a particular substring

Given two strings σ and τ of lowercase letters, say that τ is a *good subsequence* of σ if τ is obtained from σ by deleting (possibly empty) initial and final segments of σ, with two consecutive letters in τ being such that the first one is alphabetically before the second one. For instance, `u`, `aef`, `bh`, `bcd` and `de` are amongst the good subsequences of `uaeffbhbcdde`.

Write a C program that prompts the user to input a sequence of at most 20 lowercase letters on one line, ending in carriage return, and prints out the **longest, leftmost** good subsequence of the input (between two good subsequences of different length, the longest is preferred; between two good subsequences of the same length, the leftmost is preferred). The program should report an error message in case the input contains nonlowercase letters or more than 20 lowercase letters before the final new line character.

Your program should behave as follows:

```
$ a.out
Enter input: ab cd
Input is incorrect or too long.

$ a.out
Enter input: abCd
Input is incorrect or too long.

$ a.out
Enter input: abcdefghijabcdefghija
Input is incorrect or too long.

$ a.out
Enter input: abcdefghijabcdefghij
The solution is: abcdefghij

$ a.out
Enter input: aaaaa
The solution is: a

$ a.out
Enter input: uaeffbhbcdde
The solution is: aef
```

5 ☞ Finding another particular substring

Write a C program that behaves as the program in the previous question but prints out the **longest, lexicographically first** good subsequence of the input (between two good subsequences of different length, the longest is preferred; between two good subsequences of the same length, the one that comes first lexicographically is preferred).

Your program should behave as follows:

```
$ a.out
Enter input: acdaadefacdeabc
The solution is: acde
```