Algorithms and Programming
Laboratory number 05

Exercise 01

During a training session each athlete of a group of professional
cyclists is checked during each lap.
For each athlete all lap times are stored in a file with the following
format.
The first line of the file stores the number of cyclist in the group.
Then, for each cyclist, the file stores:
- On the first line, his/her name (string of 30 characters at most),
  identifier (integer value), and number of laps performed.
- On the second line, all lap times,
  time_1 time_2 ... time_N, stored as real values.

Write a program that, after reading the file and storing its content
in a proper data structure, is able to reply to the following menu
inquiry:
- list: the program prints-out the number of athletes, their names,
  identifiers, and number of laps performed.
- detail name: given an athlete name, the program prints-out
  his/her identifier, and all lap times.
- best: the program prints-out the name, identifier, all lap times
  and the average lap time for the athlete whose average lap time is
  smaller.
- stop: end the program.

Notice that all operations can be performed more than once till the
stop command is issued.

Example

Let the following be the input file:

4 zd
Rossi 100 3
1.30 1.38 1.29
Bianchi 101 5
1.46 1.43 1.42 1.51 1.28

```
Neri 117 2
1.26 1.34
Verdi 89 4
2.01 1.45 1.43 1.38
```

The following is a run example of the program:

```
Input file name: cyclist.txt
Command? list
Number of athletes : 4
Name: Rossi #Id:100 #Laps:3
Name: Bianchi #Id:101 #Laps:5
Name: Neri #Id:117 #Laps:2
Name: Verdi #Id:89 #Laps:4
Command? best
Name:Neri #Id number:117 Laps:2 Times: 1.26 1.34 (Average:1.30)
Command? details Bianchi
#Id:101 #Laps:5 Times: 1.46 1.43 1.42 1.51 1.28
Command? stop
Program ended.
```

Exercise 02

A matrix of strings is stored in a file with the following format:

```
R C
string_1_1 string_1_2 string_1_3 ... string_1_C
...
string_R_1 string_R_2 string_R_3 ... string_R_C
```

where R and C are integers, representing the number of rows and
columns of the matrix, and string_i_j are the strings stored in
the matrix.
Each string is at most 20 characters long.
Notice that strings on each row are already alphabetically ordered.

Write a C program able to "merge" all strings in the matrix in a
single array of strings, i.e., to store all strings of the matrix
in a unique array containing all strings alphabetically ordered.

Notice that:

1. the array has a number of elements equal to RxC, and that
   each of its element contains a string
2. as the strings on each row of the matrix are already ordered,
   strings do not need to be re-ordered completely to generate the
   final array.
   See the "suggestions" for further details.

The result has to be stored in an output file, which contains
the total number of strings on the first row, and all string in the
following lines.

Example

The following is an example of a correct input file:

```
4 3
milano torino venezia
bari genova taranto
firenze napoli roma
bologna cagliari palermo
```

and the following is the corresponding output file:

```
12
bari
bologna
cagliari
firenze
genova
milano
napoli
palermo
roma
taranto
torino
venezia
```

Suggestions

- The dynamic matrix has to be defined as:
  char ***mat;
  Allocate it accordingly:

- first, allocate the column of pointers to pointers to characters
- then, allocate the rows of pointers to characters
- finally, allocate the strings.

- The "strdup" function can be used to avoid string allocation.

- To merge two "already ordered" array it is possible to use a "merge" algorithm which is much more efficient than a sorting algorithm.

  Merge uses the following scheme.

  1. Let us suppose A and B are two ordered arrays that have to be merged into array C.
  2. Set a=0 and b=0 and c=0
  3. As A is sorted A[a] is the smallest element within A.
     As B is sorted B[b] is the smallest element within B.
     Then, the smallest element between A[a] and B[b] is the smallest overall, and it can be copied into array C before any other element.
     That is:

     ```
     if (A[a]<B[b]) {
        C[c]=A[a]; a++;
     } else {
        C[c]=B[b]; b++;
     }
     c++;
     ```

     book p252

  4. After that, repeat all steps from step 3, until all elements within A OR B have been copied into C.
  5. At this point, copy the remaining element of A OR B (the one with remaining elements) into C.

  Notice that merge has a linear time cost whereas sorting is linearithmic or quadratic.
  In this case, it is sufficient to "merge" all rows of the matrix (i.e., array of strings) in a single array of strings.