**SYSTEMS ARCHITECTURE**

**LAST NAME:** .....
**NAME:** .....

**LABORATORY SESSION 5: Files**

**PROBLEM 1 (10th January 2018)**
Implement a program that, for a given text file, prints the frequency of its letters.
The program:

- Will receive the name of the input file as its one and only argument. Otherwise a proper error message will be printed.
- The program will not be case sensitive (lowercases and uppercases will be equivalent)
- No special characters (as accent marked letters or ˜N) will be considered.
- The total number of letters between A and Z is 26.
- Will print the total number of letters in the file will be printed, the 10 most frequent letters ordered by its frequency, and the relative frequency of the top 5 letters
- You can use, among others, the following functions:
  - `int toupper(int c)`
  - `void qsort(void *base, size_t nmemb, size_t size, int (*compar)(const void *, const void *));`
- In the case of using `qsort`, you should define a new data structure in order to know the letter that corresponds to each frequency.

Execution sample:

```
$ ./frecuencia
./frecuencia needs 1 argument
$
$ ./frecuencia WindsOfWinter.txt
fopen: No such file or directory
$
$ ./frecuencia elquijote.txt lazarilloTormes.txt
./frecuencia needs 1 argument
$
$ ./frecuencia elquijote.txt
Total number of letters: 1599048
Letters sorted by frequency: EAOSN RLDUI
Most frequent letters:
E: 13.88 %
A: 12.09 %
O: 9.59 %
S: 7.86 %
N: 6.78 %
$
$ ./frecuencia lazarilloTormes.txt
Total number of letters: 80068
Letters sorted by frequency: EAOSN RLDUI
Most frequent letters:
E: 13.35 %
A: 13.01 %
O: 9.62 %
S: 7.03 %
N: 6.58 %
$ ./frecuencia TomSawyer_en.txt
Total number of letters: 301110
Letters sorted by frequency: ETAON HISRD
Most frequent letters:
E: 12.04 %
T: 9.75 %
A: 7.92 %
O: 7.84 %
N: 6.80 %
```

**PROBLEM 2 (16th June 2015)**
Codify a program that can receive the following arguments (in any order):

```
./palindromo nombreFichero [-num]
```

The program should:

1. Check if it has at least 1 argument and at the most 2. If not, it should exit.
2. If it receives only 1 argument, the argument should be a file.
3. If it receives 2 arguments, one should be a file and the other one the string `-num`. They are not following a given order. You can assume that the file does not start by -.
4. The program should check that the arguments are correct and that the file has written permissions.
5. If `-num` is an argument, the program will only accept numbers as input from the user.

Once checked the input arguments:

1. The program should wait a string from the user.
2. If a new line is introduced, it will return to the step 1.
3. If `-num` was an input argument, the program will check if the user input is a number. If not, it will print an error message and return to the step 1.
4. It will check if the input string (without the '\n') is palindrome (some examples: `ana`, `3223`, `dabalearrozalazorraelabad`). It will not consider `Ana` as palindrome.
5. If the string is a palindrome, it will **add** the string in a new line of the file introduced as input argument.
6. It will return to the step 1.

If the user introduces CTRL+D, the program will finish. It shall not have memory leaks.

**PROBLEM 3 (14th June 2016 and 14th January 2016)**
Implement a program that receives as input parameter the name of a file (that should exist and should have written permissions). The program accepts commands from the user until the user presses `CTRL+D`, `CTRL+C` or the user does not insert any command in 10 seconds (it should finish with no memory leaks).
The commands have the following format:
`operation numStrings string1 string2 string3...`
where `operation` should be `toupper` or `tolower`.

The program should:

- Check if it has only 1 input parameter. If not, it prints an error and finishes.
- Check if the file passed as input parameter exists and if it has written permissions. If it doesn't exist or if it has no written permissions, the program should print an error and finish.
- Parse the commands introduced by the user. If there is an error (the operation is not `toupper` or `tolower`, numStrings is not a number, there are not enough strings within the command), it should print `Not Supported`.
- Otherwise, it should convert the strings as indicated and print them out both at standard output and in the file passed as input parameter.
- The program should append to the file the lines, that is, it should not overwrite the file passed as input parameter.
- You can use the functions `int toupper(int c)` and `int tolower(int c)`

Example of execution:

```
$./transform f1
add 2 10 15
Not Supported
count
Not Supported
```

```
toupper 2 hola adios
HOLA ADIOS
tolower 2a hola adios
Not Supported
tolower 3 hola adios
Not Supported
tolower 3 hola adios BYE
hola adios bye
->No user commands in 10 seconds. Exiting
```

The content of `f1` after this execution should be:

```
HOLA ADIOS
hola adios bye
```