L:EIC / SO2122:
Processing Text and Files
(using Standard C Library functions)

Q1. Consider the following program that takes two strings from the comand line (`argv[1]` e `argv[2]`) and manipulates them with the *string* subset of the Standard C Library (`clib`) API. Compile it and try it.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAX_STR_SIZE  64

int main(int argc, char* argv[]) {
  char* p1 = (char*)malloc(MAX_STR_SIZE * sizeof(char));
  char* p2 = (char*)malloc(MAX_STR_SIZE * sizeof(char));

  int result = strcmp(argv[1], argv[2]);

  if (result == 0)
    printf("the strings are the same\n");
  else if (result < 0)
    printf("%s < %s\n", argv[1], argv[2]);
  else
    printf("%s > %s\n", argv[1], argv[2]);

  strcpy(p1, argv[1]);
  strcpy(p2, argv[2]);
  printf("p1 holds:%s\n", p1);
  printf("p2 holds:%s\n", p2);

  strcat(p1,p2);
  printf("p1 holds:%s\n", p1);
```

```
  strcat(p2,p1);
  printf("p2 holds:%s\n", p2);

  return EXIT_SUCCESS;
}
```

Run the command `man 3 string` to see the full set of functions in this API. Based on this example, write a program that:

- gets a string from the command line and transforms it into an equivalent string but in lowercase;
- gets two strings from the command line and indicates whether the first occurs within the second;
- gets two strings from the command line and indicates how many times the first occurs in the second.

Suggestion: do `man tolower` and `man toupper` to see `clib` functions that may be relevant.

Q2. Using the subset of the Standard C Library API for file manipulation (e.g., `fopen`, `fclose`, `fseek`, `fread` and `fwrite`), write a program `mycat` that:

- receive as argument the name of a file and print its content (similar to the command `cat` with 1 argument);
- receive as argument the names of several files and prints the contents of all of them, in sequence (similar to the command `cat` with multiple arguments).

```
$ cat > file1
This is a test
^D
$ cat > file2
Another test
^D
$ cat > file3
And yet another
^D
$ ./mycat file1
This is a test
$ ./mycat file1 file3
This is a test
And yet another
$ ./mycat file1 file2 file3
This is a test
Another test
And yet another
```

Q3. Create a program `chcase` that receives as arguments the name of a file and an option that sends to the standard output the content of the file as follows:

- with all characters in uppercase if the option is `-u`;
- with all characters in lowercase if the option is `-l`;
- unchanged, if no option is given.

as in the following examples:

```
$ cat > teste.txt
Ads fTsfsdsR DSda BVHGIsdssdeSds
Dfcdfd 45343f rerTEuk
qqfFGfhuymIOu 95r342
^D
$ ./chcase -u teste.txt
ADS FTSFSDSR DSDA BVHGISDSSDESDS
DFCDFD 45343F RERTEUK
QQFFGFHUYMIOU 95R342
$ ./chcase -l teste.txt
ads ftsfsdsr dsda bvhgisdssdesds
dfcdfd 45343f rerteuk
qqffgfhuymiou 95r342
$ ./chcase teste.txt
Ads fTsfsdsR DSda BVHGIsdssdeSds
Dfcdfd 45343f rerTEuk
qqfFGfhuymIOu 95r342
```

Q4. Write a program that receives the name of two files `file1` and `file2` as arguments and that copies the contents of `file1` into `file2`. If the second file does not exist, it should be created. If it exists, its contents will be overwritten. This is the usual semantics of the `cp` command of the Bash shell.

```
$ cat > file1
This is a test
^D
$ ./mycp file1 file2
$ cat file2
This is a test
$ cat > file3
Another test
^D
$ ./mycp file3 file2
$ cat file2
Another test
```

Q5. Write a program `mywc` that, given a text file as a command line argument, prints:

- the number of characters in the file, if an option `-c` is used;
- the number of words in the file, if an option `-w` is used;
- the number of lines in the file, if an option `-l` is used;
- the number of characters, if no option is given.

```
$ cat > file.txt
This is a test
^D
$ ./mywc -c file.txt
15
$ ./mywc -w file.txt
4
$ ./mywc -l file.txt
1
$ ./mywc file.txt
15
```

Compare your program to the shell command `wc`.

Q6. Write a program `mygrep` that, given a string and a file from the command line, prints all occurrences of the string in the file, indicating the line and column where these begin. The output would look something like:

```
$ ./mygrep needle haystack.txt
[2:17]
[5:2]
[23:7]
```