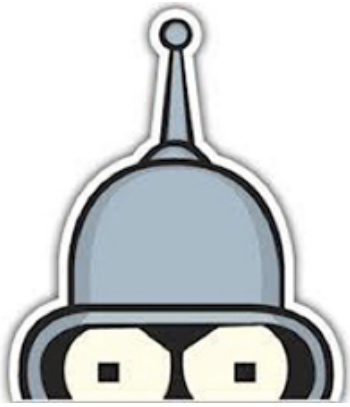

B4- Elementary Programming in C

B-CPE-330

Exam 4

Meaningful Door

v1.0

Instructions

Administrative details

- Any kind of communication is forbidden.
- You must put your student card, face up, on your desk.
- This is an exam, it is forbidden to talk, listen to music, make noise, or produce any kind of nuisance that may disturb other students or prevent this exam from running smoothly.
- Your cell phones must be OFF.
- In case of a technical problem with the subject, you have to ask a supervisor (and ONLY a supervisor).
- Any equipment that is not explicitly allowed is strictly forbidden.
- You have the right to use blank sheets of paper and a pen.
- Any exit is definitive.

Delivery

You will have to use the '~**/rendu**' directory as the root directory for your delivery.
Each exercise must be carried out in the directory mentioned in the task's header.
The '~**/rendu**' directory must contain a file called '**author**', containing your login followed by a '\n'.

Terminal

```
$> cat -e ~/rendu/author
firstname.lastname@epitech.eu$
```

EXERCISES MUST BE CARRIED OUT IN ORDER. Grading will stop as soon as an exercise is deemed false.

To make a delivery, at any time, use the **exam_rendu** command and follow the instructions.

The collection will be handled by a program, so you must respect the names, paths, of all the files and directories...

The code

- Useful fonctions are sometimes given in the '~**/sujet/misc**' directory.
- Collection of the code is automated, and a program will check that your exercices work correctly.
- Allowed functions are specified in each task's header. You can re-code any function you like.
- Any function not explicitly allowed is forbidden.

If a task mention files from the ~**/looneytunes** directory, it means that these files will be added by the autograder during the compilation.

If a task ask you to make a program, it's main function must always return 0 (even in case of failure).

All your C files must be compliant with gnu90.
Example of compilation with the gnu90 standard: gcc -std=gnu90 *.c

Task 1

show_first_param

Delivery: ~/rendu/show_first_param/*
Language: C
Compilation: make
Binary name: show_first_param
Allowed functions: write

Write a program that displays the first argument given to the program, followed by a '\n'.
If no parameters are given, you will display a '\n'.

Terminal

```
$> ./show_first_param Neque porro quisquam est qui dolorem ipsum quia dolor
Neque
$> ./show_first_param | cat -e
$
$> ./show_first_param 5 4 3 2 1
5
```

Task 2

mult_table

Delivery: ~/rendu/mult_table/*
Language: C
Compilation: make
Binary name: mult_table
Allowed functions: atoi, printf

Write a program called *mult_table* that takes a number and displays the corresponding multiplication table, the output will be followed by a '\n'.
If no parameters are given, you will display a '\n'.

Here are some examples:

Terminal

```
$> ./mult_table 9
1 x 9 = 9
2 x 9 = 18
3 x 9 = 27
4 x 9 = 36
5 x 9 = 45
6 x 9 = 54
7 x 9 = 63
8 x 9 = 72
9 x 9 = 81
$> ./mult_table 19
1 x 19 = 19
2 x 19 = 38
3 x 19 = 57
4 x 19 = 76
5 x 19 = 95
6 x 19 = 114
7 x 19 = 133
8 x 19 = 152
9 x 19 = 171
$> ./mult_table | cat -e
$
```

Task 3

hidenp

Delivery: ~/rendu/hidenp/*
Language: C
Compilation: make
Binary name: hidenp
Allowed functions: write

Write a program that takes two strings and displays *1* followed by '\n' if the first string is hidden in the second string, otherwise *0* followed by '\n'.

A string is hidden in the second, if each character from the first one can be found in the second one with the same order as in the first string.

- "fgex.;" is hidden in "tyf34gdgf;'ektufjhgdgex.;.;rtjynur6"
- "abc" is hidden in "2altrb53c.sse"
- "abc" is not hidden in "btarc"

If no parameters are given, the program displays '\n'.

Terminal

```
$> ./hidenp padinton "paqefwtdjetyiytjneytjoeyjnejeyj"
1
$> ./hidenp ddf6vewg64f gtwthgdwthdwfteewhrtag6h4ffdhsd | cat -e
0$
./hidenp | cat -e
$
```

Task 4

epur_str

Delivery: ~/rendu/epur_str/*
Language: C
Compilation: make
Binary name: epur_str
Allowed functions: write

Write a program named 'epur_str' that takes a string and displays the words separated by a single space.
The last word will be followed by a '\n' (even if there is none).
There will be no spaces before the first or after the last word.
A word is a string separated by either spaces or tabulations, or the start or the end of the string.
If there are no parameters, epur_str displays '\n'.

Terminal

```
$> ./epur_str "abc cba abc cab cba" | cat -e
abc cba abc cab cba$
$> ./epur_str "   Remus   et  Romulus   sont les deux mamelles de Rome  " | cat -e
Remus et Romulus sont les deux mamelles de Rome$
$> ./epur_str "$(echo -e "\tHello\t\t how are you?\t ")" | cat -e
Hello how are you?$
$> ./epur_str | cat -e
$
```

Task 5

count_island

Delivery: ~/rendu/count_island/*
Language: C
Compilation: make
Binary name: count_island
Allowed functions: read, write, open, close, malloc, free

Write a program called '**count_island**'.

This program takes a file containing a series of lines of equal length, containing either the dot character (.) or the 'X' character.

These lines form a rectangle of '.' containing islands of 'X's.
A line is a series of '.' and 'X' characters terminated by a '\n'.
The lines are all of the same length. The maximum size of a line is 1024 characters.
A column of characters is formed by all the characters in the file that are separated from the start of their respective lines by the same amount of characters.

Two caracters are considered touching each other if they are :

- contiguous and on the same line
  OR
- on the same column and on contiguous lines

An island of 'X' is formed by the set of characters touching each other.

The program must look through the whole file line by line and display it on the screen with all the 'X's from the islands replaced by their number in order of apparition in the file.

The program must do this computation starting by the start of the file.
There can be no two different results from the same file.
If the file is empty, an error occured or no file is passed, the program simply writes the '\n' character to the standard output.
The file contains at most 10 islands.

You can find example files in the misc directory.

Examples

Content of the 'toto' file:

```
.................XXXXXXXX.....................................................
...................XXXXXXXXX.......XXXXXXXX.....................................
.................XXXXXXXX.............XXX...XXXXX.................................
...................XXXXXX.....X...XXXXXXXXXXX...................................
.................................X.....................................................
......XXXXXXXXXXXX.............X.....................................................
...................X.............XXXXXXXXX.....................................
...................X.........XXXXXXXXXXXX.....................................
...................X.....................................................
XX.....................................................................XXXX
XX...................XXXXXXXXXXXXX.....................................X
.....................................................................
.....................................................................X.
...................XXXXX.....................................XX
```

Terminal

```
$> ./count_island toto
.................00000000.....................................................
.....................000000000.......11111111.....................
.................00000000.............111...11111.................
.....................000000.....2...11111111111.....................
.................................2.....................................
......3333333333333.............2.....................................
...................3.............222222222.....................
...................3.........222222222222.....................
...................3.....................................................
44.....................................................................5555
44...................6666666666666.....................................5
.....................................................................
.....................................................................7.
.....................88888.....................................77
$> ./count_island flop_likes
.....................................................................
...0........0.....11111......2222222...3333333333..4444444444......
...00......00....11...11....22.....22.....3333.....4444444444......
...0000..0000...11.....11...22.....22......33......44..............
...00.0000.00...11.....11...22.....22......33......44..............
...00...0..00...11.....11...22222222.......33......44444...........
...00......00...111111111...2222.............33......44444...........
...00......00..11.......11..22.22...........33......44..............
...00......00..11.......11..22...5.........33......44..............
...00......00..11.......11..22....6......333333....4444444444......
...00......00.11.........11.22.....77..3333333333..4444444444..8...
.....................................................................
$> ./count_island titi | cat -e
..0.......$
...1.2....$
....22.33.$
.4..22....$
.4..2...5.$
$> ./count_island | cat -e
$
```