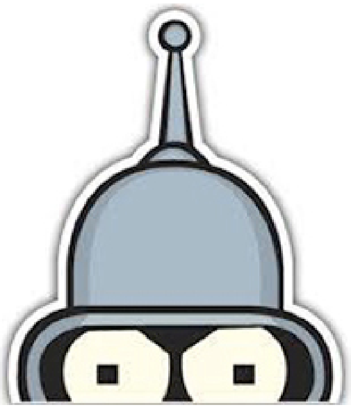

B4- Elementary Programming in C

B-CPE-330

Exam 5

Silver Obscure

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).

Task 1

union

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

Write a program that takes two strings and displays, without doubles, the characters that appears in one or the other followed by a '\n'.

The output will be in order of apparition in the command line.
If there isn't two parameters, display a '\n'.

```
Terminal
$> ./union "zpadinton" "paqefwtdjetyiytjneytjoeyjnejeyj" | cat -e
zpadintoqefwjy$
$> ./union ddf6vewg64f gtwthgdwthdwfteewhrtag6h4ffdhsd | cat -e
df6vewg4thras$
$> ./union "nothing" "this sentence hides nothing" | cat -e
nothigs ecd$
$> ./union | cat -e
$
```

Task 2

rostring

Delivery: ∼/rendu/rostring/*
Language: C
Compilation: make
Binary name: rostring
Allowed functions: write

Write a program that takes a string and displays it after doing a right to left rotation.
So, the first word becomes the last and the order of the other words is not modified.

Words are strings separated by spaces or tabulations.
Words will be displayed separated by only one space.

The output will always be followed by a '\n'.

Terminal

```
$> ./rostring "abc   " | cat -e
abc$
$> ./rostring "Let there    be more light" | cat -e
there be more light Let$
$> ./rostring "     AkjhZ zLKIJz , 23y" | cat -e
zLKIJz , 23y AkjhZ$
$> ./rostring | cat -e
$
```

Task 3

my_sort_wordtab

Delivery: ~/rendu/my_sort_wordtab/my_sort_wordtab.c
Language: C
Compilation: gcc -o my_sort_wordtab my_sort_wordtab.c ~/looneytunes/main.c
Binary name: my_sort_wordtab
Allowed functions: none

Write a function that, using **ascii order**, sorts the words received via **my_str_to_wordtab**. The sort should be executed by switching the array's pointers.
The function must be prototyped as follows:

```
int my_sort_wordtab(char **tab);
```

The function will always return 0.

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

g-diam

Delivery: ~/rendu/g-diam/*
Language: C
Compilation: make
Binary name: g-diam
Allowed functions: printf, malloc, free

If there is no parameter, the program displays '\n'.
The program takes a string that represent a series of links between cells.
Links are separated by spaces.
These cells are represented by numbers and the links by '-'.

Examples:
If the cell 2 is linked to the cell 3, representations will be either "2-3" or "3-2".

This string represents a graph. The program must display the length of its longest path.
It is not possible to go back, that is to say no cell can be used more than once. (See examples).

Terminal

```
$> ./g-diam | cat -e
$
$> ./g-diam "17-5 5-8 8-2 17-21 21-2 5-2 2-6 6-14 6-12 12-19 19-14 14-42" | cat -e
10$
$> ./g-diam "1-2 2-3 4-5 5-6 6-7 7-8 9-13 13-10 10-2 10-11 11-12 12-8 16-4 16-11 21-8 21-12
18-10 18-13 21-18" | cat -e
15$
```