Chapter II

Introduction

Regardless of the programming language considered, the `printf` function,(or its equivalents) is always highly useful. The main reason is the ease of its formatting, and the support of diverse types in variable numbers. Some variations even propose to be able to write the resulting string of characters either in a file descriptor or in a particular stream. Some also propose to recall this string without printing it. In short, undeniably, printf is a vital function. In this project, we ask you to recode it and add it to your `libft` so that you can use it in all your future projects, such as `ft_ls`...

Chapter III

Objectives

The versatility of the `printf` function in `C` represents a great exercise in programming for us. This project is of moderate difficulty. It will enable you to discover variadic functions in `C` in a particularly relevant context as well as learn about a great example of a basic "dispatcher" in `C` via the use of an array of function's pointers.

The key to a successful `ft_printf` is a well-structured and good extensible code. This is because as time goes by, you will be more and more tempted to continue to extend your `ft_printf`, making your life easier for your future projects. So take your time to code properly while keeping in mind that you will have to read again your code in a few weeks or a few months to extend its functionality according to your needs. It would be a shame not to being able to do it because your can't read your own work. What do you think?

Chapter IV

General Instructions

- Your function must be called `ft_printf`.
- Your project must be written in accordance with the Norm.
- You have to handle errors carefully. In no way can your program quit in an unexpected manner (Segmentation fault, bus error, double free, etc).
- Your must submit a Makefile which will compile a libftprintf.a. This lib will be linked to our testing main to give you your results.
- You'll have to submit a file called `author` containing your username followed by a '\n' at the root of your repository.

```
$>cat -e author
xlogin$
```

- You are allowed to use the following functions:
  - `write`
  - `malloc`
  - `free`
  - `exit`
  - The functions of man 3 `stdarg`
- You can ask your questions on the forum.

Chapter V

Mandatory part

- You have to recode the `libc`'s `printf` function.
- Your function will be called `ft_printf` and will be prototyped similarly to `printf`.
- You won't do the buffer management in the `printf` function.
- You have to manage the following conversions: `sSpdDioOuUxXcC`
- You must manage `%%`
- You must manage the flags `#0-+` and space
- You must manage the minimum field-width
- You must manage the precision
- You must manage the flags `hh`, `h`, `l`, `ll`, `j`, et `z`.

`man 3 printf / man 3 stdarg`

Chapter VI

Bonus part

Below are a few interesting ideas of bonuses for you to either create or use. You can of course add your own bonuses, which will then be evaluated directly by your correctors.

- More detailed conversions management: `eE`, `fF`, `gG`, `aA`, `n`.
- More detailed flags management: `*`, `$`, `L`, `'`.
- Non-existing flags management: `%b` to print in binary, `%r` to print a string of non-printable characters, `%k` to print a date in any ordinary ISO format etc.
- Management of alter tools for colors, fd or other fun stuff like that :)

```
printf("Le fichier{cyan}%s{eoc} contient : {red}%s{eoc}", filename, str);
```