C Piscine

Mini-project 01 : match / nmatch

Staff 42 pedago@42.fr

Summary: Second mini-project of the C Piscine @ 42. Contrary to popular belief, this subject does not contain nuts.

Contents

Chapter 1

Foreword

Want to stay awake ... Just follow these caffeine-free tips, and you'll learn how to stay awake at work!

Strut your stuff.
Studies show that taking a 20 minute walk can boost your energy levels and decrease fatigue.
Involve your ears.
Give your eyes a break.
Stretch it out.
Fuel up with healthy snacks.
When all else fails, use cold water.

Chapter 2

Instructions

- The exercises are carefully laid out in order of difficulty, from easiest to hardest. An exercise is only graded if all previous ones are correct. In other words: the grading for a day stops at the first mistake.
- Be mindful of the submission procedures indicated at the start of every exercise.
- Your exercises will be checked and graded by your fellow classmates.
- On top of that, your exercises will be checked and graded by a program called Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. Be as thorough as possible!
- Moulinette relies on a program called **norminette** to check if your files respect the Norm. An exercise containing files that do not respect the Norm will be graded 0.
- Using a forbidden function is considered cheating. Cheaters get -42, and this grade is non-negotiable.
- If **ft_putchar()** is an authorized function, we will compile your code with our **ft_putchar.c**.
- You'll only have to submit a **main()** function if we ask for a program.
- Moulinette compiles with these flags: **-Wall -Wextra -Werror**, and uses **gcc**.
- If your program doesn't compile, it will be graded 0.
- You should not leave any additional file in your directory than those specified in the subject.

norminette must be launched with the -R CheckForbiddenSourceHeader flag. Moulinette will use it too.

The forewords are entirely unrelated to the subjects and can safely be ignored.

Chapter 3

match

Turn-in directory : ex00/
Files to turn in: match.c
Allowed functions: None

- The purpose of this function is to find out whether two strings match.
- **s1** and **s2** are considered to match when **s1** and **s2** are identical.
- If **s2** contains a star ('*'), we can replace this star by any characters string (even empty) to make **s1** and **s2** identical.
- **s2** may hold as many stars as you'd like.
- For example, "**main.c**" and "***.c**" match because it is possible to replace '*' by the string "**main**" to render those two strings identical.
- Here's how it should be prototyped :

```
int         match(char *s1, char *s2);
```

- It must return **1** if **s1** and **s2** match, or **0** if they don't.

Chapter 4

nmatch

Turn-in directory : ex01/
Files to turn in: nmatch.c
Allowed functions: None

- The aim of this function is to count the amount of times two strings match.
- When we have two or more stars, multiple string combinations can be suitable.
- **nmatch** calculates the total amount of combinations.
- Here are some examples :
  - "**abcbd**" & "***b***" match twice : ["**a**","**cbd**"] and ["**abc**", "**d**"]
  - "**abc**" & "**a****" match 3 times : [**nothing**,"**bc**"], ["**b**", "**c**"] and ["**bc**", **nothing**]
- Here's how it should be prototyped :

```
int         nmatch(char *s1, char *s2);
```

- **nmatch** returns the number of combinations that match.