



C Bootcamp

Mini-project 01 : match / nmatch

Staff WeThinkCode_ info@wethinkcode.co.za

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

Contents

I	Foreword	2
II	Instructions	3
III	match	5
IV	nmatch	6

Chapter I

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 II

Instructions

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- 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. So if you want to avoid bad surprises, be as thorough as you can be.
- Moulinette is not very open-minded. It won't try and understand your code if it doesn't respect the Norm. Moulinette relies on a program called **Norminator** to check if your files respect the norm. TL;DR: it would be idiotic to submit a piece of work that doesn't pass **Norminator**'s check.
- 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.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We will not take into account a successfully completed harder exercise if an easier one is not perfectly functional.


- Moulinette compiles with these flags: `-Wall -Wextra -Werror`, and uses `gcc`.
- If your program doesn't compile, you'll get 0.
- Exercises in shell have to be executed with `/bin/sh`.
- You cannot leave any additional file in your directory than those specified in the subject.
- Got a question? Ask the neighbor on your right. Otherwise, try the neighbor on your left.
- Your reference guide is called `Google / man / the Internet /`
- Try checking out the "C Bootcamp" part of the forum on the intranet.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- By Odin, by Thor ! Use your brain !!!



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

Chapter III

match

	Exercise 00
match	
Turn-in directory : <i>ex00/</i>	
Files to turn in : match.c	
Allowed functions : None	
Notes : n/a	


- 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 IV

nmatch

	Exercise 01
nmatch	
Turn-in directory : <i>ex01/</i>	
Files to turn in : nmatch.c	
Allowed functions : None	
Notes : n/a	

- 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.