



# B1 - Unix & C Lab Seminar

B-CPE-100

## match - nmatch

Characters matching





# match - nmatch

language: C



- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.



- Don't push your `main` function into your delivery directory, we will be adding our own. Your files will be compiled adding our `main.c`.
- If one of your files prevents you from compiling with `*.c`, the Autograder will not be able to correct your work and you will receive a 0.



All `.c` files from your delivery folder will be collected and compiled with your `libmy`, which must be found in `lib/my/`. For those of you using `.h` files, they must be located in `include/` (like the `my.h` file).



The only allowed system call for this project is `write`.



Both exercises will be built individually from each other with our own main and linked with your lib.

```
Terminal
~/B-CPE-100> ./lib/my/build.sh
~/B-CPE-100> gcc -o match *.c test_files/match_main.c -I./include -L./lib/my -lmy
~/B-CPE-100> gcc -o nmatch *.c test_files/nmatch_main.c -I./include -L./lib/my -lmy
```

Your library will be built using the `lib/my/build.sh` script you previously made (see Day07).

Don't forget that you need a coherent test policy to ensure your program outputs are correct. To do so:

- split your functions in **as many small functions as possible**, so that each function is responsible for one single thing (according to the Coding Style),
- **write unit tests** to test exhaustively all of these functions.



## MATCH

---

**Delivery:** match.c

The purpose of this function is to **find out if two strings match**, that is to say when they are identical.

If the second string contains an asterisk (\*), this asterisk can be replaced with any character string (even an empty one) so that the two strings become identical.

For instance, `main.c` and `*.c` match because it is possible to replace the asterisk with the `main` string, making them identical.

This second string can contain an unlimited number of asterisks.

The function must be prototyped as follows:

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

The function returns 1 if the strings match, and 0 otherwise.

## NMATCH

---

**Delivery:** nmatch.c

The purpose of this function is to **count the number of times two strings match**.

When there are two or more asterisks, several string combinations are possible. Your function must calculate the total number of such combinations.

For instance, `abcbcd` and `*b*` match two times: `(a, cbd)` and `(abc, d)`

`abc` and `a**` match three times: `(nothing, bc)` ; `(b, c)` and `(bc, nothing)`.

The function must be prototyped as follows:

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

The function returns the number of combinations that match.