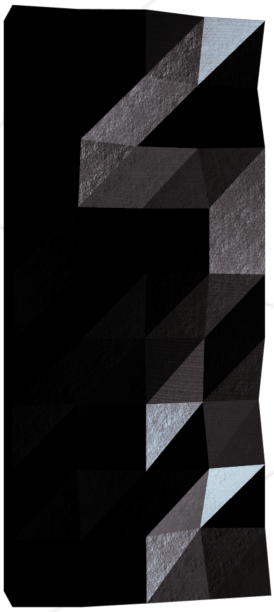

C Bootcamp

Rush 00

Staff WeThinkCode_ info@wethinkcode.co.za

Summary: This document is the subject for Rush00 of the C Bootcamp @ WeThinkCode_.

Contents

Chapter I

Instructions

- Each member of the group can register the whole group to defense.
- The group MUST be registered to defense.
- Any question concerning the subject would complicate the subject.
- You have to follow the submission procedures for all your exercises.
- This subject could change up to an hour before submission.
- 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 the following flags: -Wall -Wextra -Werror; and uses `gcc`.
- If your program doesn't compile, you'll get `0`.
- Rushs exercises have to be carried out by group of 2, 3 or 4.
- In the group_promo.txt file, you'll find the list of imposed groups with the subject.
- You must therefore do the project with the imposed team and show up at the defense slot you've selected, with all of your teammates.
- You project must be done by the time you get to defense. The purpose of defense is for you to present and explain any and all details of your work.
- Each member of your group must be fully aware of the works of the project. Should you choose to split the workload, make sure you all understand what everybody's done. During defense, you'll be asked questions, and the final grade will be based on the worst explainations.
- It goes without saying, but gathering the group is your responsibility. You've got all the means to get in contact with your teammates: phone, email, carrier pigeon,

spiritism, etc. So don't bother blurping up excuses. Life isn't always fair, that's just the way it is.

- However, if you've really tried everything one of your teammates remains unreachable : do the project anyway, and we'll try and see what we can do about it during defense. Even if the group leader is missing, you still have access to the submission directory.

- If you want bonus points, you may submit other subjects.

- 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.

Make sure the subject that was originally assigned to your group works perfectly before considering bonuses : If a bonus subject works, but the original one fails the tests, you'll get 0.

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

Chapter II

Foreword

If you understand some of those jokes, your soul might be saved !

```
I changed my password everywhere to 'incorrect.'
That way when I forget it, it always reminds me, 'Your password is incorrect.' -

Why does the computer programmer ignore the warning on the cigarette carton?
Because he's seen so many warnings he only cares about errors. -

How does a network administrator nerd greet people who come to his house?
Welcome to 127.0.0.1. -

What do you get when you put root beer in a square glass ?
Beer. -

Sodium sodium sodium sodium sodium sodium sodium sodium Batman !

There is a band named 1023MB.
They haven't had any gigs yet. -
```

But enough laughts now is time to rush.

Chapter III

Main subject

	Exercise 00
Rush0X	
Turn-in directory : $ex00/$	
Files to turn in : `main.c, ft_putchar.c, rush0X.c`	
Allowed functions : `write`	
Notes : `n/a`	

- Files to submit: `main.c`, `ft_putchar.c` and your `rush0X.c`, '0X' represents the rush number. For example `rush00.c`.
- Example of main.c :

```
int        main()
{
    rush(5, 5);
    return (0);
}
```

- You must therefore create the function `rush` taking two variables of type `int` as arguments, named respectively `x` and `y`.
- Your function `rush` should display (on-screen) a rectangle of `x` characters for width, and `y` characters for length.
- Your `main` will be modified during defense, to check if you've handled everything you're supposed to. Here's an example of test we'll perform :

```
int        main()
{
    rush(123, 42);
    return (0);
}
```

Chapter IV

Rush 00

- rush(5,3) should display :

```
$>./a.out
o---o
|   |
o---o
$>
```

- rush(5, 1) should display :

```
$>./a.out
o---o
$>
```

- rush(1, 1) should display :

```
$>./a.out
o
$>
```

- rush(1, 5) should display :

```
$>./a.out
o
|
|
|
o
$>
```

- rush(4, 4) should display :

```
$>./a.out
o--o
|  |
|  |
o--o
$>
```

Chapter V

Rush 01

- rush(5,3) should display :

```
$>./a.out
/***\
*   *
\***/
$>
```

- rush(5, 1) should display :

```
$>./a.out
/***\
$>
```

- rush(1, 1) should display :

```
$>./a.out
/
$>
```

- rush(1, 5) should display :

```
$>./a.out
/
*
*
*
\
$>
```

- rush(4, 4) should display :

```
$>./a.out
/**\
*  *
*  *
\**/
$>
```

Chapter VI

Rush 02

- rush(5,3) should display :

```
$>./a.out
ABBBA
B   B
CBBBC
$>
```

- rush(5, 1) should display :

```
$>./a.out
ABBBA
$>
```

- rush(1, 1) should display :

```
$>./a.out
A
$>
```

- rush(1, 5) should display :

```
$>./a.out
A
B
B
B
C
$>
```

- rush(4, 4) should display :

```
$>./a.out
ABBA
B  B
B  B
CBBC
$>
```

Chapter VII

Rush 03

- rush(5,3) should display :

```
$>./a.out
ABBBC
B   B
ABBBC
$>
```

- rush(5, 1) should display :

```
$>./a.out
ABBBC
$>
```

- rush(1, 1) should display :

```
$>./a.out
A
$>
```

- rush(1, 5) should display :

```
$>./a.out
A
B
B
B
A
$>
```

- rush(4, 4) should display :

```
$>./a.out
ABBC
B  B
B  B
ABBC
$>
```

Chapter VIII

Rush 04

- rush(5,3) should display :

```
$>./a.out
ABBBC
B   B
CBBBA
$>
```

- rush(5, 1) should display :

```
$>./a.out
ABBBC
$>
```

- rush(1, 1) should display :

```
$>./a.out
A
$>
```

- rush(1, 5) should display :

```
$>./a.out
A
B
B
B
C
$>
```

- rush(4, 4) should display :

```
$>./a.out
ABBC
B  B
B  B
CBBA
$>
```