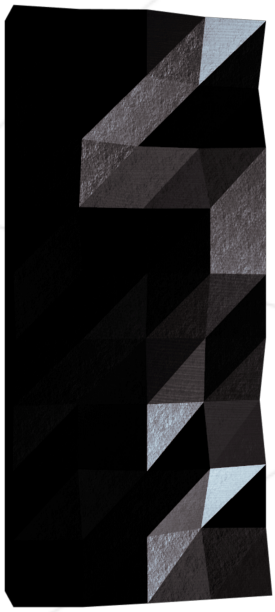

C Piscine

Rush00

Summary: This document contains the instructions for Rush00 of the C Piscine @ 42.

Version: 7.2

Contents

Chapter I

Foreword

Here are the lyrics of a famous TV show theme for everyone to enjoy!

```
[Verse 1]
I wanna be the very best
Like no one ever was
To catch them is my real test
To train them is my cause

I will travel across the land
Searching far and wide
Each pokemon to understand
The power that's inside

[Chorus]
Pokemon! Gotta catch 'em all! It's you and me
I know it's my destiny,
Pokemon! Oh you're my best friend
In a world, we must defend
Pokemon! A heart so true
Our courage will pull us through,

You teach me and I'll teach you,
Pokemon! Gotta catch'em all

[Chorus]

Every challenge along the way
With courage I will face.
I will battle every day
To claim my rightful place.
Come with me,
The time is right,
There's no better team.
Arm in arm we'll win the fight!
It's always been our dream!
```

`[Chorus]`

I bet you were singing just now! But for now, let's stay focused; this subject has nothing to do with *Pocket Monsters*!

Chapter II

Instructions

- You must do the project with the imposed team. Meeting with your teammates is part of your tasks, using any appropriate means (Slack, email, phone if available, or directly in person).
- If you have really tried everything to reach one of your teammates, but they still remain unreachable, follow the instructions from your pedagogical team, if provided. The default procedure is to do the project with the available teammates, and discuss the situation during the evaluation. Even if the group leader is missing, you still have access to the submission directory.
- Your work must comply with the Norm. If you have bonus files/functions, they are included in the norm check, and you will receive a score of 0 if there is a norm error.
- You must handle errors coherently. You may either print an error message or simply return control to the user.
- Your project must be completed and pushed to the Git repository by the deadline displayed on the project's main page on the intranet.
- You must follow the submission procedure described at the end of this document, if provided.
- Your program must compile using `cc` with the following flags: -Wall -Wextra -Werror. If there is a moulinette, it will use the same compiler and flags.
- If your program does not compile, you will receive a score of `0`.
- The group will be automatically registered for the defense. You must attend your evaluation with all of your teammates. The purpose of the defense is to present and explain your work comprehensively.
- Do not cancel your evaluation; you will not get a second one.
- Each member of the group must be fully aware of all the details of the project. If you choose to split the workload, ensure you understand every part completed by other team members. This understanding may be verified during the evaluation.
- This document contains **five** different versions of the subject.

- Your team must complete one version, determined by the following calculation:
  - Take the numerical index (from 1 to 26) of the first letter of the team leader's login.
  - Compute modulo 5 on that number.
  - The result determines which subject to complete.
- Refer to the attached `groups_subject.txt` file for more details and examples.

Bonus Points:

You can earn bonus points by submitting other versions of the project with their corresponding names. It is also possible to create a single binary accepting a command line argument to switch from one version to another.

Make sure your assigned subject works perfectly before attempting any bonuses!
If your bonus submission works, but your original assignment fails, you will receive a score of 0

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`	

Your program must display a rectangle on the screen and follow these requirements:

Files to submit:

- `main.c`
- `ft_putchar.c`
- `rush0X.c`, where "0X" represents the rush number (e.g., rush00.c).
    - The next chapters will describe the specific constraints for each rush number.

Compilation:

- These **three** files will be compiled together.
- The `ft_putchar.c` file must contain the `ft_putchar` function.

Example of `main.c` file:

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

Function requirements:

- You must write a function called `rush` with the following specifications:
  - It must take two integer arguments, named x and y.
  - The function must be placed in the rush0X.c file.
- The function must be prototyped as follows:

```
void        rush(int x, int y);
```

- Your `rush` function must display a rectangle on the screen with a width of `x` characters and a height of `y` characters.
- Your function must never crash or enter an infinite loop.
- Your `main` function will be modified during the defense to check whether you have handled all required cases.
  Here is an example of a test that will be performed:

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

Chapter IV

Rush00

- 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

Rush01

- 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

Rush02

- 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

Rush03

- 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

Rush04

- 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
$>
```

Chapter IX

Submission and peer-evaluation

Submit your assignment to your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Make sure to double-check the names of your files to ensure they are correct.

This assignment is not automatically verified by a program. You are free to organize your files as you wish, as long as you submit the required files and comply with the project requirements.

`You must submit only the files explicitly requested in the project instructions.`