RUSH01

Important information for each Rush:

1. Read carefully: subject, evaluation sheet, and all questions.
2. Do the project if you find the time (optional).
3. On the evaluation day, pick a computer in the cluster with some space in between.
4. Be an example
    a. Have a positive attitude during your evaluations
    b. Explain mistakes to the Pisciners
    c. Encourage the group
    d. Try to give meaningful tips on coding in general.
5. With these rush evaluations, we have the chance to set the tone for the right peer-2-peer experience and also set the tone for our future students.
6. Be strict with your evaluations. We all must follow the evaluation sheets and treat each group similarly. It is unfair to our Pisciners if some groups pass and others fail while fulfilling the same criteria.
7. If the evaluation does not pass a specific test it stops.
8. Check the specific information for each rush (see below).
9. After each evaluation. Take a few minutes and write down who was showing up and your assessment of the group and specific people.
10. Try to ask a code question to each piscine in the group, such as “What does this line do?” or “Why did you write this line?”.
11. If any suspicious behavior happens during the evaluation (suspicion of cheating, bad behavior…). Write to the Pedago staff about it, try to report as much as you can.
12. Remember to be kind, understanding and human. The Pisciners are going through a stressful time so don’t add “wood to the fire”.

Rush 01

Evaluations length: 60 min

- Globals have to be static or const (NORM)
- Source code will be compiled as follows: cc -Wall -Wextra -Werror -o rush01 *.c
- Folder structure must be correct. ex00/….
- No limit on files (all have to be .c)
- The evaluation sheet is, in my opinion, quite harsh and please use common sense when evaluating beginners.

> **Explanation**
>
> Ask the shyest team member to explain how the code works.
>
> If the student can't explain it, no matter how other team members reply, leave No in the scale and carry on with the evaluation.

Formal

- Errors -> “Error\n”
- Spaces between numbers | none trailing
- No additional new lines
- This is the ONLY acceptable input for your program. Any other input must be considered an error.
    - ./rush01 "col1top col2top col3top col4top col1bottom col2bottom col3bottom col4bottom row1left row2left row3left row4left row1right row2right row3right row4right"
    - Single spaces
    - Integers of the right size (right amount)(correct digits)
    - One argument

Code

- Be aware of inputs with multiple solutions
- Only use tools and testers you feel comfortable with and fully understand.
- Tests in shared directory
- Manual tests: https://www.brainbashers.com/showskyscraper.asp?date=1230&size=4&diff=1
- They do not protect their malloc functions => Fail (see below)
    - #define malloc(X) NULL
    - (ulimit -v X; ./a.out [ARGS…])

        - Ulimit can be used to (in this case) limit the amount of virtual memory a program can use. It ensures that every malloc is protected, defining the malloc to NULL only checks for the first malloc.
- They have memory leaks => Fail (see below)
    - Check with: “cc -g -fsanitize=leak”
    - Check with “cc -g ; valgrind ./a.out”
- Invalid read/write => Fail (see below)
    - Check with: “cc -g -fsanitize=address”
    - Check with “cc -g; valgrind ./a.out”
- Test used functions with “nm -u”

Three different kind of Solutions that you can find correcting rush01

Bruteforce, try all the combinations in order and check with a function looking for the correct answer

Conditional, basically use if/else if/else instructions

Recursive, solve it with a function that call itself many times

Overview

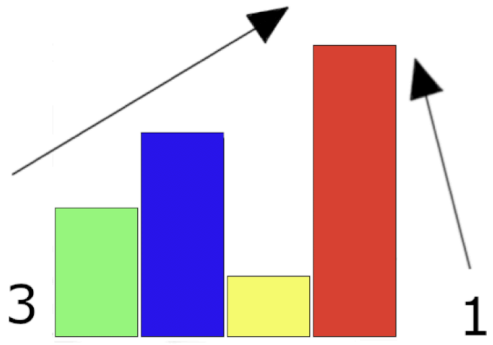
Here's how we'll launch your program :

```
> ./rush-01 "col1up col2up col3up col4up col1down col2down col3down col4down row1left row2left
    row3left row4left row1right row2right row3right row4right"
```

	col1up	col2up	col3up	col4up	
row1left					row1right
row2left					row2right
row3left					row3right
row4left					row4right
	col1down	col2down	col3down	col4down	

./rush-01 "4 3 2 1 1 2 2 2 4 3 2 1 1 2 2 2"

You will have a table with these value:



- Here's an example of intended input/output for a valid set.

```
./rush-01 "4 3 2 1 1 2 2 2 4 3 2 1 1 2 2 2" | cat -e
1 2 3 4$
2 3 4 1$
3 4 1 2$
4 1 2 3$
```

IMPORTANT!

Interesting things to talk about / explain to them if you still have time after evaluation:

- Ask them some questions to evaluate if they understood the **problem**
- Ask them to explain how they found the **solution** and why is it a good one?
- Ask them to explain what a **matrix** is and how to implement it
- Check for **complex** code or **new functions** for them like malloc and talk with them about the **theory** behind that function like for this example the memory allocation
- Make sure if they failed the rush that they understood perfectly the **reason.**