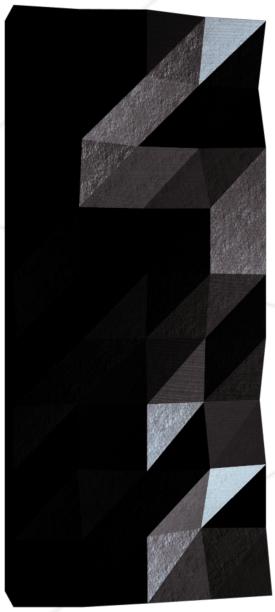

Go Piscine

Rush 01

Summary: This document is the subject for Rush01 of the Go Piscine @ 42Tokyo.

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.
- You will have to handle errors coherently. Feel free to either print an error message, or simply return control to the user.
- Rushes exercises have to be carried out by group of 2, 3 or 4.
- You must therefore do the project with the imposed team and show up at Your defense slot, 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 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 the defense, you'll be asked questions and the final grade will be based on the worst explanations.
- 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. If you have done everything but you weren't able to gather other mates, then report this issue from https://form.run/@adm-42tokyo.
- You must use the latest version of Go.
- Your turn-in directory for each exercise should look something like this:

```
ex[XX]
|-- main.go
|-- vendor
    |-- ft
        |-- printrune.go
    |-- piscine
        |-- *.go
```

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.

Chapter II

Rush01

	Exercise 00
Rush01	
Turn-in directory : *ex*00/	
Files to turn in : `*`	
Allowed packages : `os`	
Allowed builtin functions : `append, cap, copy, delete, len, make, new`	

Create a program that place some black squares('B'), so as to meet the following conditions:

- No two black squares are orthogonally adjacent.
- No group of white squares is separated from the rest of the grid by black squares.
- Each numbered cell can see precisely that many white squares(expressed by '.' character or numbers) in total by looking in all four orthogonal directions, counting itself.
- Grid will be passed by commandline arguments and the size of grid will always be 5 x 5.

```
$> go mod init ex00
$> go run . "...2." "..6.4" "5...6" "7.6.." ".3..." | cat -e
.B.2B$
..6B4$
5B..6$
7.6B.$
.3B..$
```

- You should only print one solution if the grid is solvable.
- Your function should never crash or loop indefinitely.
- Whenever an invalid arguments are given(arguments contains different characters than . and numbers, length of each argument are not equal) or a grid isn't solvable you should return an error message or prints nothing and gives back control.

```
$> go mod init ex00
$> go run . | cat -e
Error$
$> go run . "..." | cat -e
Error$
$> go run . "...A." "..6.4" "5....6" "7.6.." ".3..." | cat -e
Error$
```

Other tests:

```
$> go mod init ex00
$> go run . "....." "8.8.." ".7.7." "..8.5" "....." | cat -e
.B.B.$
8.8..$
.7.7B$
..8.5$
B.B.B$
```

```
$> go mod init ex00
$> go run . "37..." "..8.." "....." "..8.." "...69" | cat -e
37.B.$
B.8..$
...B.$
B.8..$
...69$
```

Chapter III

Bonus

	Exercise 01
Bonus	
Turn-in directory : $ex01/$	
Files to turn in : `*`	
Allowed packages : `Everything except external packages`	
Allowed builtin functions : `*`	

Add more functionality to ex00.
For example:

- Grid generation functionality.
- Visualization of the solver.
- Able to solve bigger map.

Other (creative) functionality will be graded too.
For each functionality, 5 points are given. (Max 25points)