who-are-you

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 1

required

XP

Files to submit

Allowed functions

0.00 B

who-are-you.sh

—

Instructions

"You just woke up in a dark alley... You can not remember who you are... The only thought that comes to your mind is a tag that says: subject Id: 70"

Create the file who-are-you.sh which will remind you who you are by showing your name only.

* Where to look : <https://01.tomorrow-school.ai/assets/superhero/all.json>
* What to use : curl and jq
* The output should be exactly like the example below but with the expected name

$./who-are-you.sh | cat -e

"name"$

$

Level 2

required

XP

Files to submit

Allowed functions

0.00 B

to-git-or-not-to-git.sh

—

Instructions

Write in a file to-git-or-not-to-git.sh the command that will show the name, power and gender of the superhero with the id: 170.

* Where to look : [superhero](https://01.tomorrow-school.ai/assets/superhero/all.json)

The output should be exactly like the example below:

$ bash to-git-or-not-to-git.sh

Chameleon

28

Male

$

Something

Level 2

required

XP

Files to submit

Allowed functions

0.00 B

to-git-or-not-to-git.sh

—

Instructions

Write in a file to-git-or-not-to-git.sh the command that will show the name, power and gender of the superhero with the id: 170.

* Where to look : [superhero](https://01.tomorrow-school.ai/assets/superhero/all.json)

The output should be exactly like the example below:

$ bash to-git-or-not-to-git.sh

Chameleon

28

Male

$

Level 2

required

XP

Files to submit

Allowed functions

0.00 B

mastertheLS

—

Instructions

A little voice speaks in your head:

"Now that you know who you are. You need to remember what you can do..."

The instincts are coming back...

Put in a file mastertheLS the command line that will:

* list the files and directories of the current directory.
* Ignore the hidden files, the "." and the "..".
* Separate the results with commas **only**.
* Order them by ascending order of access time (the newest first).
* Have the directories ends with a /.

Hint

Read the man...

Level 3

required

XP

Files to submit

Allowed functions

0.00 B

r

—

Instructions

"keep training ..."

Create a file r, which shows R on a line when the cat command is executed

A line is a sequence of characters preceding the [end of line](https://en.wikipedia.org/wiki/Newline) character ('\n').

Usage

$ cat -e r

R$

$

Level 3

required

XP

Files to submit

Allowed functions

0.00 B

look

—

Instructions

"start looking ..."

Create a file look, which will look for and show, in the current directory and its sub-folders:

* everything that starts with an a or,
* all the files ending with a z or,
* all files starting with z and ending with a!.

Hint

Read the find man...

cl-camp4

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 3

required

XP

Files to submit

Allowed functions

0.00 B

myfamily.sh

—

Instructions

"someone familiar"

Create a file myfamily.sh, which will show a subject's family (key: relatives).

* The quotes have to be removed.
* The subject will be decided depending on his ID which will be contained in the environment variable HERO\_ID.
* Where to look : <https://01.tomorrow-school.ai/assets/superhero/all.json>
* What to use : curl, jq and others...

Usage

$ export HERO\_ID=1

$ ./myfamily.sh

Marlo Chandler-Jones (wife); Polly (aunt); Mrs. Chandler (mother-in-law); Keith Chandler, Ray Chandler, three unidentified others (brothers-in-law); unidentified father (deceased); Jackie Shorr (alleged mother; unconfirmed)

$

cl-camp5

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 3

optional

XP

Files to submit

Allowed functions

0.00 B

lookagain.sh

—

Instructions

"keep looking..."

Create a file lookagain.sh, which will look, from the current directory and its sub-folders for:

* all the files ending with .sh.

That command will only show the name of the files without the .sh. The files will be in descending order (as shown in the below example).

Usage

$ ./lookagain.sh | cat -e

file3$

file2$

file1$

$

Hint

A little cuting might be useful...

cl-camp6

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 4

optional

XP

Files to submit

Allowed functions

0.00 B

countfiles.sh

—

Instructions

"Now, do your inventory"

Create a file countfiles.sh, which will print the number **(and only the number)** of regular files and folders contained in the current directory and its sub-folders (the current directory must be included in the count):

Usage

$ ./countfiles.sh | cat -e

12$

$

now-get-to-work

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 4

optional

XP

Files to submit

Allowed functions

0.00 B

my\_answer.sh

—

Instructions

"Something terrible happened"

clone this repo : [github.com/01-edu/the-final-cl-test](https://github.com/01-edu/the-final-cl-test)

Submit your solution in the file my\_answer.sh that will print it when executed.

Utilisation

$ bash my\_answer.sh | cat -e

John Doe$

$

Hint

"You could combine head and tails..."

explain

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 4

optional

XP

Files to submit

Allowed functions

0.00 B

explain.sh

—

Instructions

"So you want to play?"

"How did you do it?"

The commissioner thanks you for delivering the suspect. He now asks you how did you get onto his trail.

Write an explain.sh file that in this order:

* displays the first and last name of your key witness
* displays the interview number of this witness
* displays the colour and the make of the car of the main suspect
* displays the names of the 3 other main suspects that were not arrested (in alphabetical order of their last name)

Usage

$ ./explain.sh | cat -e

FirstNameOfWitness LastNameOfWitness$

123456$

Red Ferrari$

FirstNameOfSuspect1 LastNameOfSuspect1$

FirstNameOfSuspect2 LastNameOfSuspect2$

FirstNameOfSuspect3 LastNameOfSuspect3$

$

Hint

"The answers in the usage are obviously there for formatting only, but you obviously knew that already... or did you? :) "

teacher

available

Level 5

bonus

XP

Files to submit

Allowed functions

1.65 kB

teacher.sh

—

Instructions

"I need this report!"

"A **true teacher** does not jealously guard his mastery."

The commissioner was most impressed by the result of your answers. He has now asked your help to create a training program that will be based on your report for the future police inspectors.

In a training folder, various folders mystery were prepared with some key differences in the data for each group of trainees.

To simplify your life as a teacher and as the answers will not be exactly the same in each mystery folder, you need to write a teacher.sh file that does the following:

* step 1, it isolates, into an environment variable and through a similar command that you used to find it, the number (**and only the number**) of the key interview that helped solved the mystery (Unlike the interview number, the path will not change across the mystery folders that will be tested, hence the command to find the interview is the same across all the mystery folders that will be tested).
* step 2, it prints the newly created environment variable.
* step 3, it prints what the interview contains.
* step 4, it prints the content of the environment variable MAIN\_SUSPECT.

Now show them who is the boss...

printdigits

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 3

required

XP

Files to submit

Allowed functions

0.00 B

printdigits/main.go

github.com/01-edu/z01.PrintRune, --allow-builtin

Instructions

Write a program that prints the decimal digits in ascending order (from 0 to 9) on a single line.

A line is a sequence of characters preceding the end of line character ('\n').

Usage

$ go run .

0123456789

$

Notions

* [01-edu/z01](https://github.com/01-edu/z01)

isnegative

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 4

required

XP

Files to submit

Allowed functions

0.00 B

isnegative.go

github.com/01-edu/z01.PrintRune, --allow-builtin

Instructions

Write a function that prints 'T' (true) on a single line if the int passed as parameter is negative, otherwise it prints 'F' (false).

Expected function

func IsNegative(nb int) {

}

Usage

Here is a possible program to test your function :

package main

import "piscine"

func main() {

piscine.IsNegative(1)

piscine.IsNegative(0)

piscine.IsNegative(-1)

}

And its output :

$ go run .

F

F

T

$

Notions

* [01-edu/z01](https://github.com/01-edu/z01)

Something is wrong ? [Submit a](https://github.com/01-edu/public/issues/new?body=isnegative%0A---%0A&title=isnegative+subject)

# printcomb

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 4

required

XP

Files to submit

Allowed functions

0.00 B

printcomb.go

github.com/01-edu/z01.PrintRune, --no-lit=\d{3},

Instructions

Write a function that prints, in ascending order and on a single line: all **unique** combinations of three different digits so that, the first digit is lower than the second, and the second is lower than the third.

These combinations are separated by a comma and a space.

Expected function

func PrintComb() {

}

Usage

Here is a possible program to test your function :

package main

import "piscine"

func main() {

piscine.PrintComb()

}

This is the incomplete output :

$ go run . | cat -e

012, 013, 014, 015, 016, 017, 018, 019, 023, ..., 689, 789$

$

* 000 or 999 are not valid combinations because the digits are not different.
* 987 should not be shown because the first digit is not less than the second.

Notions

# **printnbr**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 4

optional

XP

Files to submit

Allowed functions

0.00 B

printnbr.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a function that prints an int passed in parameter. All possible values of type int have to go through. You cannot convert to int64.

### **Expected function**

func PrintNbr(n int) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"piscine"

"github.com/01-edu/z01"

)

func main() {

piscine.PrintNbr(-123)

piscine.PrintNbr(0)

piscine.PrintNbr(123)

z01.PrintRune('\n')

}

And its output :

$ go run .

-1230123

$

### **Notions**

* [01-edu/z01](https://github.com/01-edu/z01)
* [numeric types](https://golang.org/ref/spec#Numeric_types)

# **printcombn**

succeeded

Level 5

bonus

XP

Files to submit

Allowed functions

1.65 kB

printcombn.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

* Write a function that prints all possible combinations of **n** different digits in ascending order.
* n will be defined as : 0 < n < 10

Below are the references for the **printing format** expected.

* (for n = 1) '0, 1, 2, 3, ..., 8, 9'
* (for n = 3) '012, 013, 014, 015, 016, 017, 018, 019, 023,...689, 789'

### **Expected function**

func PrintCombN(n int) {

}

### **Usage**

Here is a possible program to test your function :

package main

import "piscine"

func main() {

piscine.PrintCombN(1)

piscine.PrintCombN(3)

piscine.PrintCombN(9)

}

And its output :

$ go run .

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

012, 013, 014, 015, 016, 017, 018, ... 679, 689, 789

012345678, 012345679, ..., 123456789

$

Be mindful of your program efficiency to avoid timeouts.

### **Notions**

* [01-edu/z01](https://github.com/01-edu/z01)

# **pointone**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 4

required

XP

Files to submit

Allowed functions

0.00 B

pointone.go

--allow-builtin

### **Instructions**

* Write a function that takes a **pointer to an**int as argument and gives to this int the value of 1.

### **Expected function**

func PointOne(n \*int) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

n := 0

piscine.PointOne(&n)

fmt.Println(n)

}

And its output :

$ go run .

1

$

### **Notions**

* [Pointers](https://golang.org/ref/spec#Pointer_types)

# **ultimatepointone**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 4

required

XP

Files to submit

Allowed functions

0.00 B

ultimatepointone.go

--allow-builtin

### **Instructions**

* Write a function that takes a **pointer to a pointer to a pointer to an**int as argument and gives to this int the value of 1.

### **Expected function**

func UltimatePointOne(n \*\*\*int) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

a := 0

b := &a

n := &b

piscine.UltimatePointOne(&n)

fmt.Println(a)

}

And its output :

$ go run .

1

$

### **Notions**

* [Pointers](https://golang.org/ref/spec#Pointer_types)

# **divmod**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 4

required

XP

Files to submit

Allowed functions

0.00 B

divmod.go

--allow-builtin

### **Instructions**

* Write a function that will be formatted as below.

### **Expected function**

func DivMod(a int, b int, div \*int, mod \*int) {

}

* This function will divide the int **a** and **b**.
* The result of this division will be stored in the int pointed by **div**.
* The remainder of this division will be stored in the int pointed by **mod**.

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

a := 13

b := 2

var div int

var mod int

piscine.DivMod(a, b, &div, &mod)

fmt.Println(div)

fmt.Println(mod)

}

And its output :

$ go run .

6

1

$

### **Notions**

* [Pointers](https://golang.org/ref/spec#Pointer_types)

# **ultimatedivmod**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 5

required

XP

Files to submit

Allowed functions

0.00 B

ultimatedivmod.go

--allow-builtin

### **Instructions**

Create the following function.

### **Expected function**

func UltimateDivMod(a \*int, b \*int) {

}

UltimateDivMod should divide the dereferenced value of a by the dereferenced value of b.

* Store the result of the division in the int which a points to.
* Store the remainder of the division in the int which b points to.

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

a := 13

b := 2

piscine.UltimateDivMod(&a, &b)

fmt.Println(a)

fmt.Println(b)

}

And its output :

$ go run .

6

1

$

### **Notions**

# **printstr**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 5

required

XP

Files to submit

Allowed functions

0.00 B

printstr.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

* Write a function that prints one by one the characters of a string on the screen.

### **Expected function**

func PrintStr(s string) {

}

### **Hints**

Here is a possible program to test your function :

package main

import "piscine"

func main() {

piscine.PrintStr("Hello World!")

}

And its output :

go run . | cat -e

Hello World!

### **Notions**

* [01-edu/z01](https://github.com/01-edu/z01)

# **strlen**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 5

required

XP

Files to submit

Allowed functions

0.00 B

strlen.go

--allow-builtin

### **Instructions**

* Write a function that counts the runes of a string and that returns that count.

### **Expected function**

func StrLen(s string) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

l := piscine.StrLen("Hello World!")

fmt.Println(l)

}

And its output :

$ go run .

12

$

# **swap**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 5

required

XP

Files to submit

Allowed functions

0.00 B

swap.go

--allow-builtin

### **Instructions**

* Write a function that takes two **pointers to an**int (\*int) and swaps their contents.

### **Expected function**

func Swap(a \*int, b \*int) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

a := 0

b := 1

piscine.Swap(&a, &b)

fmt.Println(a)

fmt.Println(b)

}

And its output :

$ go run .

1

0

$

### **Notions**

* [Pointers](https://golang.org/ref/spec#Pointer_types)

# **strrev**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 6

required

XP

Files to submit

Allowed functions

0.00 B

strrev.go

--allow-builtin

### **Instructions**

* Write a function that reverses a string.
* This function will **return** the reversed string.

### **Expected function**

func StrRev(s string) string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

s := "Hello World!"

s = piscine.StrRev(s)

fmt.Println(s)

}

And its output :

$ go run .

!dlroW olleH

$

# **basicatoi**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 6

optional

XP

Files to submit

Allowed functions

0.00 B

basicatoi.go

--allow-builtin

### **Instructions**

* Write a function that simulates the behaviour of the Atoi function in Go. Atoi transforms a number defined as a string into a number defined as an int.
* Atoi returns 0 if the string is not considered as a valid number. For this exercise **only valid** string will be tested. They will only contain one or several digits as characters.
* For this exercise the handling of the signs + or - does not have to be taken into account.
* This function will **only** have to return the int. For this exercise the error return of Atoi is not required.

### **Expected function**

func BasicAtoi(s string) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.BasicAtoi("12345"))

fmt.Println(piscine.BasicAtoi("0000000012345"))

fmt.Println(piscine.BasicAtoi("000000"))

}

And its output :

$ go run .

12345

12345

0

# **basicatoi2**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 6

optional

XP

Files to submit

Allowed functions

0.00 B

basicatoi2.go

--allow-builtin

### **Instructions**

* Write a function that simulates the behaviour of the Atoi function in Go. Atoi transforms a number defined as a string in a number defined as an int.
* Atoi returns 0 if the string is not considered as a valid number. For this exercise **non-valid**string**chains will be tested**. Some will contain non-digits characters.
* For this exercise the handling of the signs + or - does not have to be taken into account.
* This function will **only** have to return the int. For this exercise the error return of Atoi is not required.

### **Expected function**

func BasicAtoi2(s string) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.BasicAtoi2("12345"))

fmt.Println(piscine.BasicAtoi2("0000000012345"))

fmt.Println(piscine.BasicAtoi2("012 345"))

fmt.Println(piscine.BasicAtoi2("Hello World!"))

}

And its output :

$ go run .

12345

12345

0

0

$

# **atoi**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 6

optional

XP

Files to submit

Allowed functions

0.00 B

atoi.go

--allow-builtin

### **Instructions**

* Write a function that simulates the behaviour of the Atoi function in Go. Atoi transforms a number represented as a string in a number represented as an int.
* Atoi returns 0 if the string is not considered as a valid number. For this exercise **non-valid**string**chains will be tested**. Some will contain non-digits characters.
* For this exercise the handling of the signs + or - **does have** to be taken into account.
* This function will **only** have to return the int. For this exercise the error result of Atoi is not required.

### **Expected function**

func Atoi(s string) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.Atoi("12345"))

fmt.Println(piscine.Atoi("0000000012345"))

fmt.Println(piscine.Atoi("012 345"))

fmt.Println(piscine.Atoi("Hello World!"))

fmt.Println(piscine.Atoi("+1234"))

fmt.Println(piscine.Atoi("-1234"))

fmt.Println(piscine.Atoi("++1234"))

fmt.Println(piscine.Atoi("--1234"))

}

And its output :

$ go run .

12345

12345

0

0

1234

-1234

0

0

$

### **Notions**

* [strconv/Atoi](https://golang.org/pkg/strconv/#Atoi)

# **sortintegertable**

succeeded

Level 7

bonus

XP

Files to submit

Allowed functions

2.10 kB

sortintegertable.go

--allow-builtin

### **Instructions**

* Write a function that reorders a slice of int in ascending order.

### **Expected function**

func SortIntegerTable(table []int) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

s := []int{5,4,3,2,1,0}

piscine.SortIntegerTable(s)

fmt.Println(s)

}

And its output :

$ go run .

[0 1 2 3 4 5]

$

4

# **iterativefactorial**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 9

required

XP

Files to submit

Allowed functions

0.00 B

iterativefactorial.go

--allow-builtin

### **Instructions**

Write an **iterative** function that returns the factorial of the int passed as parameter.

Errors (non possible values or overflows) will return 0.

### **Expected function**

func IterativeFactorial(nb int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

arg := 4

fmt.Println(piscine.IterativeFactorial(arg))

}

And its output :

$ go run .

24

$

# **recursivefactorial**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 9

required

XP

Files to submit

Allowed functions

0.00 B

recursivefactorial.go

-no-for

### **Instructions**

Write a **recursive** function that returns the factorial of the int passed as parameter.

Errors (non possible values or overflows) will return 0.

for is **forbidden** for this exercise.

### **Expected function**

func RecursiveFactorial(nb int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

arg := 4

fmt.Println(piscine.RecursiveFactorial(arg))

}

$ go run .

24

$

# **iterativepower**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 10

required

XP

Files to submit

Allowed functions

0.00 B

iterativepower.go

--allow-builtin

### **Instructions**

Write an **iterative** function that returns the value of nb to the power of power.

Negative powers will return 0. Overflows do **not** have to be dealt with.

### **Expected function**

func IterativePower(nb int, power int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.IterativePower(4, 3))

}

And its output :

$ go run .

64

$

# **recursivepower**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 10

required

XP

Files to submit

Allowed functions

0.00 B

recursivepower.go

-no-for

### **Instructions**

Write a **recursive** function that returns the value of nb to the power of power.

Negative powers will return 0. Overflows do **not** have to be dealt with.

for is **forbidden** for this exercise.

### **Expected function**

func RecursivePower(nb int, power int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.RecursivePower(4, 3))

}

And its output :

$ go run .

64

$

# **fibonacci**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 10

required

XP

Files to submit

Allowed functions

0.00 B

fibonacci.go

--allow-builtin

### **Instructions**

Write a **recursive** function that returns the value at the position index in the fibonacci sequence.

The first value is at index 0.

The sequence starts this way: 0, 1, 1, 2, 3 etc...

A negative index will return -1.

for is **forbidden** for this exercise.

### **Expected function**

package piscine

func Fibonacci(index int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

arg1 := 4

fmt.Println(piscine.Fibonacci(arg1))

}

And its output :

$ go run .

3

$

# **sqrt**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 10

optional

XP

Files to submit

Allowed functions

0.00 B

sqrt.go

--allow-builtin

### **Instructions**

Write a function that returns the square root of the int passed as parameter, if that square root is a whole number. Otherwise it returns 0.

### **Expected function**

func Sqrt(nb int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.Sqrt(4))

fmt.Println(piscine.Sqrt(3))

}

And its output :

$ go run .

2

0

# **isprime**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 10

optional

XP

Files to submit

Allowed functions

0.00 B

isprime.go

--allow-builtin

### **Instructions**

Write a function that returns true if the int passed as parameter is a prime number. Otherwise it returns false.

The function must be optimized in order to avoid time-outs with the tester.

(We consider that only positive numbers can be prime numbers)

(We also consider that 1 is **not** a prime number)

### **Expected function**

func IsPrime(nb int) bool {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.IsPrime(5))

fmt.Println(piscine.IsPrime(4))

}

And its output :

$ go run .

true

false

$

# **findnextprime**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 10

required

XP

Files to submit

Allowed functions

0.00 B

findnextprime.go

--allow-builtin

### **Instructions**

Write a function that returns the first prime number that is equal or superior to the int passed as parameter.

The function must be optimized in order to avoid time-outs with the tester.

(We consider that only positive numbers can be prime numbers)

### **Expected function**

func FindNextPrime(nb int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.FindNextPrime(5))

fmt.Println(piscine.FindNextPrime(4))

}

And its output :

$ go run .

5

5

$

# **eightqueens**

succeeded

Level 11

bonus

XP

Files to submit

Allowed functions

3.50 kB

eightqueens.go

github.com/01-edu/z01.PrintRune, -cast, len, --no-lit=\\n[0-9]

### **Instructions**

Write a function that prints the solutions to the [eight queens puzzle](https://en.wikipedia.org/wiki/Eight_queens_puzzle).

Recursivity must be used to solve this problem.

It should print something like this :

$ go run .

15863724

16837425

17468253

...

Each solution will be on a single line. The index of the placement of a queen starts at 1. It reads from left to right and each digit is the position for each column. The solutions will be printed in ascending order.

### **Expected function**

package piscine

func EightQueens() {

}

5

# **firstrune**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 10

required

XP

Files to submit

Allowed functions

0.00 B

firstrune.go

--allow-builtin

### **Instructions**

Write a function that returns the first rune of a string.

### **Expected function**

func FirstRune(s string) rune {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"github.com/01-edu/z01"

"piscine"

)

func main() {

z01.PrintRune(piscine.FirstRune("Hello!"))

z01.PrintRune(piscine.FirstRune("Salut!"))

z01.PrintRune(piscine.FirstRune("Ola!"))

z01.PrintRune('\n')

}

And its output :

$ go run .

HSO

$

# **lastrune**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 10

required

XP

Files to submit

Allowed functions

0.00 B

lastrune.go

--allow-builtin

### **Instructions**

Write a function that returns the last rune of a string.

### **Expected function**

func LastRune(s string) rune {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"github.com/01-edu/z01"

"piscine"

)

func main() {

z01.PrintRune(piscine.LastRune("Hello!"))

z01.PrintRune(piscine.LastRune("Salut!"))

z01.PrintRune(piscine.LastRune("Ola!"))

z01.PrintRune('\n')

}

And its output :

$ go run .

!!!

$

# **nrune**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 10

required

XP

Files to submit

Allowed functions

0.00 B

nrune.go

--allow-builtin

### **Instructions**

Write a function that returns the nth rune of a string. If not possible, it returns 0.

### **Expected function**

func NRune(s string, n int) rune {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"github.com/01-edu/z01"

"piscine"

)

func main() {

z01.PrintRune(piscine.NRune("Hello!", 3))

z01.PrintRune(piscine.NRune("Salut!", 2))

z01.PrintRune(piscine.NRune("Bye!", -1))

z01.PrintRune(piscine.NRune("Bye!", 5))

z01.PrintRune(piscine.NRune("Ola!", 4))

z01.PrintRune('\n')

}

And its output :

$ go run .

la!

$

# **compare**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 11

required

XP

Files to submit

Allowed functions

0.00 B

compare.go

--allow-builtin

### **Instructions**

Write a function that behaves like the Compare function.

### **Expected function**

func Compare(a, b string) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.Compare("Hello!", "Hello!"))

fmt.Println(piscine.Compare("Salut!", "lut!"))

fmt.Println(piscine.Compare("Ola!", "Ol"))

}

And its output :

$ go run .

0

-1

1

$

# **alphacount**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 11

required

XP

Files to submit

Allowed functions

0.00 B

alphacount.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a function that counts the letters of a string and returns the count.

The letters are from the Latin alphabet list only, any other characters, symbols or empty spaces shall not be counted.

### **Expected function**

func AlphaCount(s string) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

s := "Hello 78 World! 4455 /"

nb := piscine.AlphaCount(s)

fmt.Println(nb)

}

And its output :

$ go run .

10

$

# **index**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 11

required

XP

Files to submit

Allowed functions

0.00 B

index.go

--allow-builtin

### **Instructions**

Write a function that behaves like the Index function.

### **Expected function**

func Index(s string, toFind string) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.Index("Hello!", "l"))

fmt.Println(piscine.Index("Salut!", "alu"))

fmt.Println(piscine.Index("Ola!", "hOl"))

}

And its output :

$ go run .

2

1

-1

$

# **concat**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 11

required

XP

Files to submit

Allowed functions

0.00 B

concat.go

--allow-builtin

### **Instructions**

Write a function that returns the concatenation of two string passed in arguments.

### **Expected function**

func Concat(str1 string, str2 string) string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.Concat("Hello!", " How are you?"))

}

And its output :

$ go run .

Hello! How are you?

$

# **isupper**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 11

required

XP

Files to submit

Allowed functions

0.00 B

isupper.go

--allow-builtin

### **Instructions**

Write a function that returns true if the string passed as parameter contains only uppercase characters, otherwise, returns false.

### **Expected function**

func IsUpper(s string) bool {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.IsUpper("HELLO"))

fmt.Println(piscine.IsUpper("HELLO!"))

}

And its output :

$ go run .

true

false

$

# **islower**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 11

required

XP

Files to submit

Allowed functions

0.00 B

islower.go

--allow-builtin

### **Instructions**

Write a function that returns true if the string passed as the parameter contains only lowercase characters, otherwise, returns false.

### **Expected function**

func IsLower(s string) bool {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.IsLower("hello"))

fmt.Println(piscine.IsLower("hello!"))

}

And its output :

$ go run .

true

false

$

# **isalpha**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 11

required

XP

Files to submit

Allowed functions

0.00 B

isalpha.go

--allow-builtin

### **Instructions**

Write a function that returns true if the string passed as the parameter only contains alphanumerical characters or is empty, otherwise, and returns false.

### **Expected function**

func IsAlpha(s string) bool {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.IsAlpha("Hello! How are you?"))

fmt.Println(piscine.IsAlpha("HelloHowareyou"))

fmt.Println(piscine.IsAlpha("What's this 4?"))

fmt.Println(piscine.IsAlpha("Whatsthis4"))

}

And its output :

$ go run .

false

true

false

true

$

# **isnumeric**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 12

required

XP

Files to submit

Allowed functions

0.00 B

isnumeric.go

--allow-builtin

### **Instructions**

Write a function that returns true if the string passed as a parameter contains only numerical characters, otherwise, returns false.

### **Expected function**

func IsNumeric(s string) bool {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.IsNumeric("010203"))

fmt.Println(piscine.IsNumeric("01,02,03"))

}

And its output :

$ go run .

true

false

$

# **isprintable**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 12

required

XP

Files to submit

Allowed functions

0.00 B

isprintable.go

--allow-builtin

### **Instructions**

Write a function that returns true if the string passed as a parameter contains only printable characters, otherwise, returns false.

### **Expected function**

func IsPrintable(s string) bool {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.IsPrintable("Hello"))

fmt.Println(piscine.IsPrintable("Hello\n"))

}

And its output :

$ go run .

true

false

$

# **toupper**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 12

required

XP

Files to submit

Allowed functions

0.00 B

toupper.go

--allow-builtin

### **Instructions**

Write a function that capitalizes each letter in a string.

### **Expected function**

func ToUpper(s string) string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.ToUpper("Hello! How are you?"))

}

And its output :

$ go run .

HELLO! HOW ARE YOU?

$

### **Notions**

* [strings/ToUpper](https://golang.org/pkg/strings/#ToUpper)

# **tolower**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 12

required

XP

Files to submit

Allowed functions

0.00 B

tolower.go

--allow-builtin

### **Instructions**

Write a function that lower cases for each letter in a string.

### **Expected function**

func ToLower(s string) string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.ToLower("Hello! How are you?"))

}

And its output :

$ go run .

hello! how are you?

$

### **Notions**

* [strings/ToLower](https://golang.org/pkg/strings/#ToLower)

Something is wrong ? [Submi](https://github.com/01-edu/public/issues/new?body=tolower%0A---%0A&title=tolower+subject)

# **printnbrinorder**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 12

required

XP

Files to submit

Allowed functions

0.00 B

printnbrinorder.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a function which prints the digits of an int passed in parameter in ascending order. All possible values of type int have to go through, excluding negative numbers. Conversion to int64 is not allowed.

### **Expected function**

func PrintNbrInOrder(n int) {

}

### **Usage**

Here is a possible program to test your function :

package main

import "piscine"

func main() {

piscine.PrintNbrInOrder(321)

piscine.PrintNbrInOrder(0)

piscine.PrintNbrInOrder(321)

}

And its output :

$ go run . | cat -e

1230123$

$

### **Notions**

* [01-edu/z01](https://github.com/01-edu/z01)
* [rune-literals](https://golang.org/ref/spec#Rune_literals)

# **trimatoi**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 12

required

XP

Files to submit

Allowed functions

0.00 B

trimatoi.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

* Write a function that transforms numbers within a string, into an int.
* If the - sign is encountered before any number it should determine the sign of the returned int.
* This function should **only** return an int. In the case of an invalid input, the function should return 0.
* **Note**: There will never be more than one sign in a string in the tests.

### **Expected function**

func TrimAtoi(s string) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.TrimAtoi("12345"))

fmt.Println(piscine.TrimAtoi("str123ing45"))

fmt.Println(piscine.TrimAtoi("012 345"))

fmt.Println(piscine.TrimAtoi("Hello World!"))

fmt.Println(piscine.TrimAtoi("sd+x1fa2W3s4"))

fmt.Println(piscine.TrimAtoi("sd-x1fa2W3s4"))

fmt.Println(piscine.TrimAtoi("sdx1-fa2W3s4"))

fmt.Println(piscine.TrimAtoi("sdx1+fa2W3s4"))

}

And its output :

$ go run .

12345

12345

12345

0

1234

-1234

1234

1234

$

# **capitalize**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 12

required

XP

Files to submit

Allowed functions

0.00 B

capitalize.go

--allow-builtin

### **Instructions**

Write a function that capitalizes the first letter of each word **and** lowercases the rest.

A word is a sequence of **alphanumeric** characters.

### **Expected function**

func Capitalize(s string) string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.Capitalize("Hello! How are you? How+are+things+4you?"))

}

And its output :

$ go run .

Hello! How Are You? How+Are+Things+4you?

$

# **basicjoin**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

optional

XP

Files to submit

Allowed functions

0.00 B

basicjoin.go

--allow-builtin

### **Instructions**

Write a function that returns a concatenated string from the 'strings' passed as arguments.

### **Expected function**

func BasicJoin(elems []string) string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

elems := []string{"Hello!", " How", " are", " you?"}

fmt.Println(piscine.BasicJoin(elems))

}

And its output :

$ go run .

Hello! How are you?

$

### **Notions**

* [string concatenation](https://golang.org/ref/spec#Arithmetic_operators)

Something is wrong ? [Submit an iss](https://github.com/01-edu/public/issues/new?body=basicjoin%0A---%0A&title=basicjoin+subject)

# **join**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

optional

XP

Files to submit

Allowed functions

0.00 B

join.go

--allow-builtin

### **Instructions**

Write a function that returns the concatenation of all the strings of a slice of strings **separated** by the separator passed as the argument sep.

### **Expected function**

func Join(strs []string, sep string) string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

toConcat := []string{"Hello!", " How", " are", " you?"}

fmt.Println(piscine.Join(toConcat, ":"))

}

And its output :

$ go run .

Hello!: How: are: you?

$

### **Notions**

* [strings/Join](https://golang.org/pkg/strings/#Join)

# **printnbrbase**

succeeded

Level 13

bonus

XP

Files to submit

Allowed functions

3.45 kB

printnbrbase.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a function that prints an int in a string base passed as parameters.

If the base is not valid, the function prints NV (Not Valid):

Validity rules for a base :

* A base must contain at least 2 characters.
* Each character of a base must be unique.
* A base should not contain + or - characters.

The function has to manage negative numbers. (as shown in the example)

### **Expected function**

func PrintNbrBase(nbr int, base string) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"github.com/01-edu/z01"

"piscine"

)

func main() {

piscine.PrintNbrBase(125, "0123456789")

z01.PrintRune('\n')

piscine.PrintNbrBase(-125, "01")

z01.PrintRune('\n')

piscine.PrintNbrBase(125, "0123456789ABCDEF")

z01.PrintRune('\n')

piscine.PrintNbrBase(-125, "choumi")

z01.PrintRune('\n')

piscine.PrintNbrBase(125, "aa")

z01.PrintRune('\n')

}

And its output :

$ go run .

125

-1111101

7D

-uoi

NV

$

### **Notions**

* [01-edu/z01](https://github.com/01-edu/z01)

# **atoibase**

succeeded

Level 13

bonus

XP

Files to submit

Allowed functions

3.45 kB

atoibase.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a function that takes two arguments:

* s: a numeric string in a given [base](https://simple.wikipedia.org/wiki/Base_(mathematics)).
* base: a string representing all the different digits that can represent a numeric value.

And return the integer value of s in the given base.

If the base is not valid it returns 0.

Validity rules for a base :

* A base must contain at least 2 characters.
* Each character of a base must be unique.
* A base should not contain + or - characters.

String number must contain only elements that are in base.

Only valid string numbers will be tested.

The function **does not have** to manage negative numbers.

### **Expected function**

func AtoiBase(s string, base string) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.AtoiBase("125", "0123456789"))

fmt.Println(piscine.AtoiBase("1111101", "01"))

fmt.Println(piscine.AtoiBase("7D", "0123456789ABCDEF"))

fmt.Println(piscine.AtoiBase("uoi", "choumi"))

fmt.Println(piscine.AtoiBase("bbbbbab", "-ab"))

}

And its output :

$ go run .

125

125

125

125

0

$

Something is wrong ? [Submit an issue](https://github.com/01-edu/public/issues/new?body=atoibase%0A---%0A&title=atoibase+subject) or even better [propos](https://github.com/01-edu/public/edit/master/subjects/atoibase/README.md)

6

# **printprogramname**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 12

required

XP

Files to submit

Allowed functions

0.00 B

printprogramname/main.go

github.com/01-edu/z01.PrintRune, os.Args, --allow-builtin

### **Instructions**

Write a **program** that prints the name of the program.

Example of output :

student/piscine/printprogramname$ go build main.go

student/piscine/printprogramnane$ ./main

main

student/piscine/printprogramname$ go build

student/piscine/printprogramname$ ./printprogramname | cat -e

printprogramname$

student/piscine/printprogramname$ go build -o Nessy

student/piscine/printprogramname$ ./Nessy

Nessy

student/piscine/printprogramname$

# **printparams**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

required

XP

Files to submit

Allowed functions

0.00 B

printparams/main.go

github.com/01-edu/z01.PrintRune, os.Args, --allow-builtin

### **Instructions**

Write a **program** that prints the arguments received in the command line.

Example of output :

$ go run . choumi is the best cat

choumi

is

the

best

cat

$

# **revparams**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

required

XP

Files to submit

Allowed functions

0.00 B

revparams/main.go

github.com/01-edu/z01.PrintRune, os.Args, .., --allow-builtin

### **Instructions**

Write a **program** that prints the arguments received in the command line in reverse order.

Example of output :

$ go run . choumi is the best cat

cat

best

the

is

choumi

$

# **sortparams**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

optional

XP

Files to submit

Allowed functions

0.00 B

sortparams/main.go

github.com/01-edu/z01.PrintRune, os.Args, .., --allow-builtin

### **Instructions**

Write a **program** that prints the arguments received in the command line in ASCII order.

Example of output :

$ go run . 1 a 2 A 3 b 4 C

1

2

3

4

A

C

a

b

$

# **nbrconvertalpha**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

optional

XP

Files to submit

Allowed functions

0.00 B

nbrconvertalpha/main.go

os.\*, github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a **program** that prints the corresponding letter in the n position of the latin alphabet, where n is each argument received.

For example 1 matches a, 2 matches b, etc. If n does not match a valid position of the alphabet or if the argument is not an integer, the **program** should print a space (" ").

A flag --upper should be implemented. When used, the program prints the result in upper case. The flag will always be the first argument.

### **Usage**

$ go run .

$ go run . 8 5 12 12 15 | cat -e

hello$

$ go run . 12 5 7 5 14 56 4 1 18 25 | cat -e

legen dary$

$ go run . 32 86 h | cat -e

$

$ go run . --upper 8 5 25

HEY$

$

# **flags**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

optional

XP

Files to submit

Allowed functions

0.00 B

flags/main.go

os.\*, fmt.\*, github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a **program** that can take --insert (or -i), --order (or -o) and a string as arguments.

This program should :

* Insert the string given to the --insert (or -i), into the argument string, if given.
* If the flag --order (or -o) is given, order the string argument (in ASCII order).
* If there are no arguments or if the flag --help (or -h) is given, the options should be printed as in the example.
  + The short flag will have two spaces before the (-).
  + The explanation of the flag will have a tab followed by a space before the beginning of the sentence (This flag...).

Don't mind the extra spaces or tabs on the following example as they were placed there to provide a better understanding and visualization of the output in the terminal. Follow the rules above for the spacing.

Example of output :

$ go run . --insert=4321 --order asdad

1234aadds

$ go run . --insert=4321 asdad

asdad4321

$ go run . asdad

asdad

$ go run . --order 43a21

1234a

$ go run .

--insert

-i

This flag inserts the string into the string passed as argument.

--order

-o

This flag will behave like a boolean, if it is called it will order the argument.

$ go run . -h

--insert

-i

This flag inserts the string into the string passed as argument.

--order

-o

This flag will behave like a boolean, if it is called it will order the argument.

$ go run . --help

--insert

-i

This flag inserts the string into the string passed as argument.

--order

-o

This flag will behave like a boolean, if it is called it will order the argument.

# **rotatevowels**

succeeded

Level 13

bonus

XP

Files to submit

Allowed functions

4.03 kB

rotatevowels/main.go

os.\*, github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a **program** that checks the arguments for vowels.

* If the arguments contain vowels (for this exercise y is not considered a vowel) the program has to **"mirror"** the position of the vowels in the arguments (see the examples).
* If the number of arguments is less than 1, the program displays a new line ("\n").
* If the arguments do not have any vowels, the program just prints the arguments.

Example of output :

$ go run . "Hello World" | cat -e

Hollo Werld$

$ go run . "HEllO World" "problem solved"

Hello Werld problom sOlvEd

$ go run . "str" "shh" "psst"

str shh psst

$ go run . "happy thoughts" "good luck"

huppy thooghts guod lack

$ go run . "aEi" "Ou"

uOi Ea

$ go run .

$

7

# **appendrange**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

required

XP

Files to submit

Allowed functions

0.00 B

appendrange.go

append

### **Instructions**

Write a function that takes an int min and an int max as parameters. The function should return a slice of ints with all the values between the min and max.

Min is included, and max is excluded.

If min is greater than or equal to max, a nil slice is returned.

make is not allowed for this exercise.

### **Expected function**

func AppendRange(min, max int) []int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.AppendRange(5, 10))

fmt.Println(piscine.AppendRange(10, 5))

}

And its output :

$ go run .

[5 6 7 8 9]

[]

$

# **makerange**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

required

XP

Files to submit

Allowed functions

0.00 B

makerange.go

make

### **Instructions**

Write a function that takes an int min and an int max as parameters. The function must return a slice of ints with all the values between min and max.

Min is included, and max is excluded.

If min is greater than or equal to max, a nil slice is returned.

append is not allowed for this exercise.

### **Expected function**

func MakeRange(min, max int) []int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.MakeRange(5, 10))

fmt.Println(piscine.MakeRange(10, 5))

}

And its output :

$ go run .

[5 6 7 8 9]

[]

$

# **concatparams**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

required

XP

Files to submit

Allowed functions

0.00 B

concatparams.go

make, --allow-builtin

### **Instructions**

Write a function that takes the arguments received in parameters and returns them as a string. The string is the result of all the arguments concatenated with a newline (\n) between.

### **Expected function**

func ConcatParams(args []string) string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

test := []string{"Hello", "how", "are", "you?"}

fmt.Println(piscine.ConcatParams(test))

}

And its output :

$ go run .

Hello

how

are

you?

$

# **splitwhitespaces**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

required

XP

Files to submit

Allowed functions

0.00 B

splitwhitespaces.go

--allow-builtin

### **Instructions**

Write a function that separates the words of a string and puts them in a string slice.

The separators are spaces, tabs and newlines.

### **Expected function**

func SplitWhiteSpaces(s string) []string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Printf("%#v\n", piscine.SplitWhiteSpaces("Hello how are you?"))

}

And its output :

$ go run .

[]string{"Hello", "how", "are", "you?"}

$

# **printwordstables**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

optional

XP

Files to submit

Allowed functions

0.00 B

printwordstables.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a function that receives a string slice and prints each element of the slice in a seperate line.

### **Expected function**

func PrintWordsTables(a []string) {

}

### **Usage**

Here is a possible program to test your function :

package main

import "piscine"

func main() {

a := piscine.SplitWhiteSpaces("Hello how are you?")

piscine.PrintWordsTables(a)

}

And its output :

$ go run .

Hello

how

are

you?

$

# **split**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 13

optional

XP

Files to submit

Allowed functions

0.00 B

split.go

make, --allow-builtin

### **Instructions**

Write a function that receives a string and a separator and returns a slice of strings that results of splitting the string s by the separator sep.

### **Expected function**

func Split(s, sep string) []string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

s := "HelloHAhowHAareHAyou?"

fmt.Printf("%#v\n", piscine.Split(s, "HA"))

}

And its output :

$ go run .

[]string{"Hello", "how", "are", "you?"}

$

# **convertbase**

succeeded

Level 14

bonus

XP

Files to submit

Allowed functions

3.67 kB

convertbase.go

--allow-builtin

### **Instructions**

Write a function that receives three arguments:

* nbr: A string representing a numberic value in a [base](https://simple.wikipedia.org/wiki/Base_(mathematics)).
* baseFrom: A string representing the base nbr it's using.
* baseTo: A string representing the base nbr should be represented in the returned value.

Only valid bases will be tested.

Negative numbers will not be tested.

### **Expected function**

func ConvertBase(nbr, baseFrom, baseTo string) string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

result := piscine.ConvertBase("101011", "01", "0123456789")

fmt.Println(result)

}

And its output :

$ go run .

43

$

8

# **boolean**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 17

required

XP

Files to submit

Allowed functions

0.00 B

boolean/main.go

github.com/01-edu/z01.PrintRune, os.Args, .., --allow-builtin

### **Instructions**

Create a new directory called boolean.

* The code below must be copied into a file called main.go inside of the boolean directory.
* The necessary changes must be applied for the program to work.

### **Code to be copied**

func printStr(s string) {

for \_, r := range s {

z01.PrintRune(r)

}

z01.PrintRune('\n')

}

func isEven(nbr int) boolean {

if even(nbr) == 1 {

return yes

} else {

return no

}

}

func main() {

if isEven(lengthOfArg) == 1 {

printStr(EvenMsg)

} else {

printStr(OddMsg)

}

}

### **Usage**

$ go run . "not" "odd"

I have an even number of arguments

$ go run . "not even"

I have an odd number of arguments

# **point**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 17

required

XP

Files to submit

Allowed functions

0.00 B

point/main.go

github.com/01-edu/z01.PrintRune#4, --no-lit=[1-9]

### **Instructions**

Create a new directory called point.

* The code below must be copied into a file called main.go inside the point directory.
* The necessary changes must be applied so that the program works.
* The function setPoint() must work with int.

### **Code to be copied**

func setPoint(ptr \*point) {

ptr.x = 42

ptr.y = 21

}

func main() {

points := &point{}

setPoint(points)

fmt.Printf("x = %d, y = %d\n",points.x, points.y)

}

### **Usage**

$ go run .

x = 42, y = 21

$

# **displayfile**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 17

required

XP

Files to submit

Allowed functions

0.00 B

displayfile/main.go

fmt.\*, os.\*, io.\*, .., --allow-builtin

### **Instructions**

Write a program that displays, on the standard output, the content of a file given as argument.

### **Usage :**

$ go run .

File name missing

$ echo 'Almost there!!' > quest8.txt

$ go run . quest8.txt main.go

Too many arguments

$ go run . quest8.txt

Almost there!!

# **cat**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 18

optional

XP

Files to submit

Allowed functions

0.00 B

cat/main.go

.., github.com/01-edu/z01.PrintRune, os.\*, io.\*, string, --allow-builtin

### **Instructions**

Write a program that behaves like a simplified cat command.

* The options do not have to be handled.
* If the program is called without arguments it should take the standard input (stdin) and print it back on the standard output (stdout).

$ echo '"Programming is a skill best acquired by practice and example rather than from books" by Alan Turing' > quest8.txt

$ cat <<EOF> quest8T.txt

"Alan Mathison Turing was an English mathematician, computer scientist, logician, cryptanalyst. Turing was highly influential in the development of theoretical computer science, providing a formalisation of the concepts of algorithm and computation with the Turing machine, which can be considered a model of a general-purpose computer. Turing is widely considered to be the father of theoretical computer science and artificial intelligence."

EOF

$ go run . abc

ERROR: open abc: no such file or directory

exit status 1

$ go run . quest8.txt

"Programming is a skill best acquired by pratice and example rather than from books" by Alan Turing

$ go run . quest8.txt abc

"Programming is a skill best acquired by pratice and example rather than from books" by Alan Turing

ERROR: open abc: No such file or directory

$ cat quest8.txt | ./cat

"Programming is a skill best acquired by pratice and example rather than from books" by Alan Turing

$ go run .

Hello

Hello

^C

$ go run . quest8.txt quest8T.txt

"Programming is a skill best acquired by pratice and example rather than from books" by Alan Turing

"Alan Mathison Turing was an English mathematician, computer scientist, logician, cryptanalyst. Turing was highly influential in the development of theoretical computer science, providing a formalisation of the concepts of algorithm and computation with the Turing machine, which can be considered a model of a general-purpose computer. Turing is widely considered to be the father of theoretical computer science and artificial intelligence."

$

# **ztail**

succeeded

Level 18

bonus

XP

Files to submit

Allowed functions

4.58 kB

ztail/main.go

os.\*, fmt.Printf, .., --allow-builtin

### **Instructions**

Write a program that behaves like a simplified tail command that takes at least one file as an argument.

The only option to be handled is -c and will be used in all the tests as the first argument, with positive values.

For this program the os package can be used.

Handle the errors by returning a non-zero exit status but process all the files.

If several files are given, print a newline and the file name between each one of them (see below).

### **Usage**

If file1.txt & file2.txt contains :

abcdefghijklmnopqrstuvwxyz

**Note** that the files above end with a new line.

Normal cases :

$ go run . -c 4 file1.txt

xyz

$ go run . -c 4 file1.txt file2.txt

==> file1.txt <==

xyz

==> file2.txt <==

xyz

$

Error cases :

$ go run . -c 4 file1.txt nonexisting1.txt file2.txt nonexisting2.txt

==> file1.txt <==

xyz

open nonexisting1.txt: no such file or directory

==> file2.txt <==

xyz

open nonexisting2.txt: no such file or directory

$ echo $?

1

$

9

# **foreach**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 18

required

XP

Files to submit

Allowed functions

0.00 B

foreach.go

--allow-builtin

### **Instructions**

Write a function ForEach that, for an int slice, applies a function on each element of that slice.

### **Expected function**

func ForEach(f func(int), a []int) {

}

### **Usage**

Here is a possible program to test your function :

package main

import "piscine"

func main() {

a := []int{1, 2, 3, 4, 5, 6}

piscine.ForEach(piscine.PrintNbr, a)

}

And its output :

$ go run .

123456

$

### **Notions**

* [Function literals](https://golang.org/ref/spec#Function_literals)
* [Function declaration](https://golang.org/ref/spec#Function_declarations)
* [Function types](https://golang.org/ref/spec#Function_types)

# **map**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 18

required

XP

Files to submit

Allowed functions

0.00 B

map.go

make, --allow-builtin

### **Instructions**

Write a function Map that, for an int slice, applies a function of this type func(int) bool on each element of that slice and returns a slice of all the return values.

### **Expected function**

func Map(f func(int) bool, a []int) []bool {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

a := []int{1, 2, 3, 4, 5, 6}

result := piscine.Map(piscine.IsPrime, a)

fmt.Println(result)

}

And its output :

$ go run .

[false true true false true false]

$

### **Notions**

* [Function literals](https://golang.org/ref/spec#Function_literals)
* [Function declaration](https://golang.org/ref/spec#Function_declarations)
* [Function types](https://golang.org/ref/spec#Function_types)

# **any**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 18

required

XP

Files to submit

Allowed functions

0.00 B

any.go

--allow-builtin

### **Instructions**

Write a function Any that returns true, for a string slice :

* if, when that string slice is passed through an f function, at least one element returns true.

### **Expected function**

func Any(f func(string) bool, a []string) bool {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

a1 := []string{"Hello", "how", "are", "you"}

a2 := []string{"This", "is", "4", "you"}

result1 := piscine.Any(piscine.IsNumeric, a1)

result2 := piscine.Any(piscine.IsNumeric, a2)

fmt.Println(result1)

fmt.Println(result2)

}

And its output :

$ go run .

false

true

$

# **countif**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 18

required

XP

Files to submit

Allowed functions

0.00 B

countif.go

--allow-builtin

### **Instructions**

Write a function CountIf that returns, the number of elements of a string slice, for which the f function returns true.

### **Expected function**

func CountIf(f func(string) bool, tab []string) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

tab1 := []string{"Hello", "how", "are", "you"}

tab2 := []string{"This","1", "is", "4", "you"}

answer1 := piscine.CountIf(piscine.IsNumeric, tab1)

answer2 := piscine.CountIf(piscine.IsNumeric, tab2)

fmt.Println(answer1)

fmt.Println(answer2)

}

And its output :

$ go run .

0

2

$

### **Notions**

* [Function literals](https://golang.org/ref/spec#Function_literals)
* [Function declaration](https://golang.org/ref/spec#Function_declarations)
* [Function types](https://golang.org/ref/spec#Function_types)

Something is wrong ? [Submit an](https://github.com/01-edu/public/issues/new?body=countif%0A---%0A&title=countif+subject)

# **issorted**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 18

required

XP

Files to submit

Allowed functions

0.00 B

issorted.go

--allow-builtin

### **Instructions**

Write a function IsSorted() that returns true, if the slice of int is sorted, otherwise returns false.

* The function passed as an argument func(a, b int) returns a positive int if the first argument is greater than the second argument, it returns 0 if they are equal and it returns a negative int otherwise.
* To do your testing you have to write your own f function.

### **Expected function**

func IsSorted(f func(a, b int) int, a []int) bool {

}

### **Usage**

Here is a possible program to test your function (without f):

package main

import (

"fmt"

)

func main() {

a1 := []int{0, 1, 2, 3, 4, 5}

a2 := []int{0, 2, 1, 3}

result1 := IsSorted(f, a1)

result2 := IsSorted(f, a2)

fmt.Println(result1)

fmt.Println(result2)

}

And its output:

$ go run .

true

false

$

### **Notions**

* [Function literals](https://golang.org/ref/spec#Function_literals)
* [Function declaration](https://golang.org/ref/spec#Function_declarations)
* [Function types](https://golang.org/ref/spec#Function_types)

# **doop**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 18

optional

XP

Files to submit

Allowed functions

0.00 B

doop/main.go

.., os.\*, --allow-builtin

### **Instructions**

Write a program that is called doop.

The program has to be used with three arguments:

* A value
* An operator, one of : +, -, /, \*, %
* Another value

In case of an invalid operator, value, number of arguments or an overflow, the programs prints nothing.

The program has to handle the modulo and division operations by 0 as shown on the output examples below.

### **Usage**

$ go run .

$ go run . 1 + 1 | cat -e

2

$

$ go run . hello + 1

$ go run . 1 p 1

$ go run . 1 / 0 | cat -e

No division by 0

$

$ go run . 1 % 0 | cat -e

No modulo by 0

$

$ go run . 9223372036854775807 + 1

$ go run . -9223372036854775809 - 3

$ go run . 9223372036854775807 "\*" 3

$ go run . 1 "\*" 1

1

$ go run . 1 "\*" -1

-1

$

### **Notions**

* [Numeric Types](https://golang.org/ref/spec#Numeric_types)
* [Arithmetic Operators](https://golang.org/ref/spec#Arithmetic_operators)

# **sortwordarr**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 18

optional

XP

Files to submit

Allowed functions

0.00 B

sortwordarr.go

--allow-builtin

### **Instructions**

Write a function SortWordArr that sorts by ascii (in ascending order) a string slice.

### **Expected function**

func SortWordArr(a []string) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

result := []string{"a", "A", "1", "b", "B", "2", "c", "C", "3"}

piscine.SortWordArr(result)

fmt.Println(result)

}

And its output :

$ go run .

[1 2 3 A B C a b c]

$

# **advancedsortwordarr**

succeeded

Level 18

bonus

XP

Files to submit

Allowed functions

4.58 kB

advancedsortwordarr.go

.., --allow-builtin

### **Instructions**

Write a function AdvancedSortWordArr that sorts a slice of string, based on the function f passed in parameter.

### **Expected function**

func AdvancedSortWordArr(a []string, f func(a, b string) int) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

result := []string{"a", "A", "1", "b", "B", "2", "c", "C", "3"}

piscine.AdvancedSortWordArr(result, piscine.Compare)

fmt.Println(result)

}

And its output :

$ go run .

[1 2 3 A B C a b c]

$

### **Notions**

* [Function literals](https://golang.org/ref/spec#Function_literals)
* [Function declaration](https://golang.org/ref/spec#Function_declarations)
* [Function types](https://golang.org/ref/spec#Function_types)

Something is wrong ? [Subm](https://github.com/01-edu/public/issues/new?body=advancedsortwordarr%0A---%0A&title=advancedsortwordarr+subject)

10

# **rot14**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 23

required

XP

Files to submit

Allowed functions

0.00 B

rot14.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a function rot14 that returns the string within the parameter transformed into a rot14 string. Each letter will be replaced by the letter 14 spots ahead in the alphabetical order.

* 'z' becomes 'n' and 'Z' becomes 'N'. The case of the letter stays the same.

### **Expected function**

func Rot14(s string) string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"piscine"

"github.com/01-edu/z01"

)

func main() {

result := piscine.Rot14("Hello! How are You?")

for \_, r := range result {

z01.PrintRune(r)

}

z01.PrintRune('\n')

}

And its output :

$ go run .

Vszzc! Vck ofs Mci?

$

# **descendcomb**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 23

required

XP

Files to submit

Allowed functions

0.00 B

descendcomb.go

github.com/01-edu/z01.PrintRune#6, --allow-builtin

### **Instructions**

Write a function that prints in descending order and on a single line all possible combinations of two different two-digit numbers.

These combinations are separated by a comma and a space.

### **Expected function**

func DescendComb() {

}

### **Usage**

Here is a possible program to test your function:

package main

import "piscine"

func main() {

piscine.DescendComb()

}

This is the incomplete output:

$ go run . | cat -e

99 98, 99 97, 99 96, 99 95, 99 94, ..., 03 01, 03 00, 02 01, 02 00, 01 00$

# **unmatch**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 23

required

XP

Files to submit

Allowed functions

0.00 B

unmatch.go

--allow-builtin

### **Instructions**

Write a function, Unmatch, that returns the element of the slice that does not have a correspondent pair.

* If all the number have a correspondent pair, it should return -1.

### **Expected function**

func Unmatch(a []int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

a := []int{1, 2, 3, 1, 2, 3, 4}

unmatch := piscine.Unmatch(a)

fmt.Println(unmatch)

}

And its output :

$ go run .

4

$

# **fooddeliverytime**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 23

required

XP

Files to submit

Allowed functions

0.00 B

fooddeliverytime.go

—

### **Instructions**

Given the following menu with the corresponding times that each item takes to cook (burger takes 15 min, chips takes 10 min and nuggets takes 12 min), return the time that each order item takes to be prepared and the total amount of time for the order to be ready assuming the items are cooked one after the other.

Write a function FoodDeliveryTime() that takes a string and returns an int.

* Use structs to answer the subject.
* If any of the order items don't exist in the menu above return the error message 404.

### **Expected function**

type food struct {

preptime int

}

func FoodDeliveryTime(order string) int {

}

### **Usage**

Here is a possible program to test your function:

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.FoodDeliveryTime("burger"))

fmt.Println(piscine.FoodDeliveryTime("chips"))

fmt.Println(piscine.FoodDeliveryTime("nuggets"))

fmt.Println(piscine.FoodDeliveryTime("burger") + piscine.FoodDeliveryTime("chips") + piscine.FoodDeliveryTime("nuggets"))

}

And its output:

$ go run . | cat -e

15$

10$

12$

37$

# **collatzcountdown**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 23

required

XP

Files to submit

Allowed functions

0.00 B

collatzcountdown.go

--allow-builtin

### **Instructions**

Write a function, CollatzCountdown, that returns the number of steps necessary to reach 1 using the [collatz countdown](https://en.wikipedia.org/wiki/Collatz_conjecture).

* It must return -1 if start is equal to 0 or negative.

### **Expected function**

func CollatzCountdown(start int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

steps := piscine.CollatzCountdown(12)

fmt.Println(steps)

}

And its output :

$ go run .

9

$

# **abort**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 23

required

XP

Files to submit

Allowed functions

0.00 B

abort.go

.., --allow-builtin

### **Instructions**

Write a function that returns the median of five int arguments.

### **Expected function**

func Abort(a, b, c, d, e int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

middle := piscine.Abort(2, 3, 8, 5, 7)

fmt.Println(middle)

}

And its output :

$ go run .

5

$

# **shoppingsummarycounter**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 23

required

XP

Files to submit

Allowed functions

0.00 B

shoppingsummarycounter.go

--allow-builtin

### **Instructions**

You have a receipt from the grocery store and you want to know how many of each item you bought. Write a function that returns this summary.

Given a string count the total amount of times each item appears in it and return a summary including the item and its number of appearances as an int.

### **Expected function**

func ShoppingSummaryCounter(str string) map[string]int {

}

### **Usage**

Here is a possible program to test your function:

package main

import (

"fmt"

"piscine"

)

func main() {

summary := "Burger Water Carrot Coffee Water Water Chips Carrot Carrot Burger Carrot Water"

for index, element := range piscine.ShoppingSummaryCounter(summary) {

fmt.Println(index, "=>", element)

}

}

And its output:

$ go run . | cat -e

Burger => 2$

Water => 4$

Carrot => 4$

Coffee => 1$

Chips => 1$

# **compact**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

compact.go

make, --allow-builtin

### **Instructions**

Write a function Compact that takes a pointer to a slice of strings as the argument. This function must:

* Return the number of elements with [non-zero value](https://tour.golang.org/basics/12).
* Compact, i.e., delete the elements with zero-values in the slice.

### **Expected functions**

func Compact(ptr \*[]string) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

const N = 6

func main() {

a := make([]string, N)

a[0] = "a"

a[2] = "b"

a[4] = "c"

for \_, v := range a {

fmt.Println(v)

}

fmt.Println("Size after compacting:", piscine.Compact(&a))

for \_, v := range a {

fmt.Println(v)

}

}

And its output :

$ go run .

a

b

c

Size after compacting: 3

a

b

c

$

# **dealapackofcards**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

dealapackofcards.go

github.com/01-edu/z01.PrintRune, fmt.Printf

### **Instructions**

Deal a pack of 12 cards evenly between 4 players, Player 1, Player 2, Player 3 and Player 4.

Write a function DealAPackOfCards() that takes the argument, deck, as a slice of int and prints the desired output.

* Each player must be printed in a different line.

### **Expected function**

func DealAPackOfCards (deck []int) {

}

### **Usage**

Here is a possible program to test your function:

package main

import "piscine"

func main() {

deck := []int{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

piscine.DealAPackOfCards(deck)

}

And its output:

$ go run . | cat -e

Player 1: 1, 2, 3$

Player 2: 4, 5, 6$

Player 3: 7, 8, 9$

Player 4: 10, 11, 12$

$

Something is wrong ? [Submit an issue](https://github.com/01-edu/public/issues/new?body=dealapackofcards%0A---%0A&title=dealapackofcards+subject) or e

# **jumpover**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

jumpover.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a function JumpOver() that takes a string and returns another string with every third character.

* Prints the output followed by newline \n.
* If the string is empty, return newline \n.
* If there is no third character, return newline \n.

### **Expected function**

func JumpOver(str string) string {

}

### **Usage**

Here is a possible program to test your function:

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Print(piscine.JumpOver("1010101010"))

fmt.Print(piscine.JumpOver(""))

fmt.Print(piscine.JumpOver("t w e l v e"))

fmt.Print(piscine.JumpOver("12"))

}

And its output:

$ go run . | cat -e

101$

$

w v$

$

# **stringtointslice**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

stringtointslice.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a function StringToIntSlice() that takes a string and returns the corresponding int slice.

### **Expected function**

func StringToIntSlice(str string) []int {

}

### **Usage**

Here is a possible program to test your function:

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.StringToIntSlice("A quick brown fox jumps over the lazy dog"))

fmt.Println(piscine.StringToIntSlice("Converted this string into an int"))

fmt.Println(piscine.StringToIntSlice("hello THERE"))

}

And its output:

$ go run . | cat -e

[65 32 113 117 105 99 107 32 98 114 111 119 110 32 102 111 120 32 106 117 109 112 115 32 111 118 101 114 32 116 104 101 32 108 97 122 121 32 100 111 103]$

[67 111 110 118 101 114 116 101 100 32 116 104 105 115 32 115 116 114 105 110 103 32 105 110 116 111 32 97 110 32 105 110 116]$

[104 101 108 108 111 32 84 72 69 82 69]$

# **join**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

join.go

--allow-builtin

### **Instructions**

Write a function that returns the concatenation of all the strings of a slice of strings **separated** by the separator passed as the argument sep.

### **Expected function**

func Join(strs []string, sep string) string {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

toConcat := []string{"Hello!", " How", " are", " you?"}

fmt.Println(piscine.Join(toConcat, ":"))

}

And its output :

$ go run .

Hello!: How: are: you?

$

# **enigma**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

enigma.go

--allow-builtin

### **Instructions**

Write a function called Enigma that receives pointers as arguments and move its values around to hide them.

This function will put :

* a into c.
* c into d.
* d into b.
* b into a.

### **Expected function**

func Enigma(a \*\*\*int, b \*int, c \*\*\*\*\*\*\*int, d \*\*\*\*int) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

x := 5

y := &x

z := &y

a := &z

w := 2

b := &w

u := 7

e := &u

f := &e

g := &f

h := &g

i := &h

j := &i

c := &j

k := 6

l := &k

m := &l

n := &m

d := &n

fmt.Println(\*\*\*a)

fmt.Println(\*b)

fmt.Println(\*\*\*\*\*\*\*c)

fmt.Println(\*\*\*\*d)

piscine.Enigma(a, b, c, d)

fmt.Println("After using Enigma")

fmt.Println(\*\*\*a)

fmt.Println(\*b)

fmt.Println(\*\*\*\*\*\*\*c)

fmt.Println(\*\*\*\*d)

}

And its output :

$ go run .

5

2

7

6

After using Enigma

2

6

5

7

$

# **descendappendrange**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

descendappendrange.go

append

### **Instructions**

* Write a function that takes an int max and an int min as parameters. The function should return a slice of ints with all the values between the max and min.
* The max must be included, and min must be excluded.
* If max is inferior than or equal to min, return an empty slice.
* make() is not allowed for this exercise.

### **Expected function**

func DescendAppendRange(max, min int) []int {

}

### **Usage**

Here is a possible program to test your function:

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.DescendAppendRange(10, 5))

fmt.Println(piscine.DescendAppendRange(5, 10))

}

And its output:

$ go run . | cat -e

[10 9 8 7 6]

[]

$

# **shoppinglistsort**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

shoppinglistsort.go

len

### **Instructions**

You were sent to the supermarket with a shopping list. To make your shopping faster, write a function ShoppingListSort() that takes a slice of strings and sorts it, according to the string length, returning a slice in which the strings appear in ascending order.

* Strings within the input slice must be of different lengths.

### **Expected function**

func ShoppingListSort(slice []string) []string {

}

### **Usage**

Here is a possible program to test your function:

package main

import (

"fmt"

"piscine"

)

func main() {

slice := []string{"Pineapple", "Honey", "Mushroom", "Tea", "Pepper", "Milk"}

fmt.Println(piscine.ShoppingListSort(slice))

}

And its output:

$ go run . | cat -e

[Tea Milk Honey Pepper Mushroom Pineapple]$

Something is wrong ? [Submit an issue](https://github.com/01-edu/public/issues/new?body=shoppinglistsort%0A---%0A&title=shoppinglistsort+subject) or even better [propose a change !](https://github.com/01-edu/public/edit/master/subjects/shoppinglistsort/README.md)

# **comcheck**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

comcheck/main.go

os.Args, .., fmt.Println, --allow-builtin

### **Instructions**

Write a program comcheck that displays on the standard output Alert!!! followed by newline ('\n') if at least one of the arguments passed in parameter matches the string:

* 01, galaxy or galaxy 01.
* If none of the parameters match, the program displays nothing.

### **Usage**

$ go run . "I" "Will" "Enter" "the" "galaxy"

Alert!!!

$ go run . "galaxy 01" "do" "you" "hear" "me"

Alert!!!

$

# **reversemenuindex**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

reversemenuindex.go

make, --allow-builtin

### **Instructions**

The restaurant employees are having a really tough day and are delivering the customers' food in the wrong order. You need to fix the problem so that they can deliver it correctly.

Write a function ReverseMenuIndex() that takes a slice of strings as an argument and returns another slice of strings with the menu in the correct order.

* append() is not allowed for this exercise.

### **Expected function**

func ReverseMenuIndex(menu []string) []string {

}

### **Usage**

Here is a possible program to test your function:

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.ReverseMenuIndex([]string{"desserts", "mains", "drinks", "starters"}))

}

And its output:

$ go run . | cat -e

[starters drinks mains desserts]$

# **podiumposition**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

podiumposition.go

len

### **Instructions**

A F1 race just finished and the commentator is calling the finishing positions incorrectly. Help to fix this before the contestants arrive at the podium by providing the commentator with the correct podium position.

Write a function PodiumPosition that takes a slice of slices of type string and returns the competitor positions correctly.

### **Expected function**

func PodiumPosition(podium [][]string) [][]string {

}

### **Usage**

Here is a possible program to test your function:

package main

import (

"fmt"

"piscine"

)

func main() {

p4 := []string{"4th Place"}

p3 := []string{"3rd Place"}

p2 := []string{"2nd Place"}

p1 := []string{"1st Place"}

position := [][]string{p4, p3, p2, p1}

fmt.Println(piscine.PodiumPosition(position))

}

And its output:

$ go run . | cat -e

[[1st Place] [2nd Place] [3rd Place] [4th Place]]$

# **activebits**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

activebits.go

--allow-builtin

### **Instructions**

Write a function, ActiveBits, that returns the number of active bits (bits with the value 1) in the binary representation of an integer number.

### **Expected function**

func ActiveBits(n int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.ActiveBits(7))

}

And its output :

$ go run .

3

$

# **pilot**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

pilot/main.go

fmt.\*, .., ., --allow-builtin

### **Instructions**

* Create a directory called pilot.
* Inside the directory pilot create a file main.go.
* Copy the code below to main.go and add the code needed so that the program compiles.

Note: You can only add code, not delete.

### **Usage**

package main

import "fmt"

func main() {

var donnie Pilot

donnie.Name = "Donnie"

donnie.Life = 100.0

donnie.Age = 24

donnie.Aircraft = AIRCRAFT1

fmt.Println(donnie)

}

const AIRCRAFT1 = 1

# **fixthemain**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

fixthemain/main.go

github.com/01-edu/z01.PrintRune, --no-lit=\\nis

### **Instructions**

Fix the following program.

### **Program to fix**

package piscine

func PrintStr(s string) {

for \_, r := range s {

z01.PrintRune(r)

}

}

func CloseDoor(ptrDoor \*Door) bool {

PrintStr("Door Closing...")

state = CLOSE

return true

}

func IsDoorOpen(Door Door) {

PrintStr("is the Door opened ?")

return Door.state = OPEN

}

func IsDoorClose(ptrDoor \*Door) bool {

PrintStr("is the Door closed ?")

}

func main() {

door := &Door{}

OpenDoor(door)

if IsDoorClose(door) {

OpenDoor(door)

}

if IsDoorOpen(door) {

CloseDoor(door)

}

if door.state == OPEN {

CloseDoor(door)

}

}

Something is wrong ? [Submit an issue](https://github.com/01-edu/public/issues/new?body=fixthemain%0A---%0A&title=fixthemain+subject)

# **max**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 24

required

XP

Files to submit

Allowed functions

0.00 B

max.go

--allow-builtin

### **Instructions**

Write a function Max that will return the maximum value in a slice of integers. If the slice is empty it will return 0.

### **Expected function**

func Max(a []int) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

a := []int{23, 123, 1, 11, 55, 93}

max := piscine.Max(a)

fmt.Println(max)

}

And its output :

$ go run .

123

$

Something is wrong ? [Subm](https://github.com/01-edu/public/issues/new?body=max%0A---%0A&title=max+subject)

# **rockandroll**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

required

XP

Files to submit

Allowed functions

0.00 B

rockandroll.go

--allow-builtin

### **Instructions**

Write a function called RockAndRoll() that takes an int and returns a string.

* If the number is divisible by 2, print rock followed by a newline \n.
* If the number is divisible by 3, print roll followed by a newline \n.
* If the number is divisible by 2 and 3, print rock and roll followed by a newline \n.
* If the number is negative return error: number is negative followed by a newline \n.
* If the number is non divisible return error: non divisible followed by a newline \n.

### **Expected function**

func RockAndRoll(n int) string {

}

### **Usage**

Here is a possible program to test your function:

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Println(piscine.RockAndRoll(4))

fmt.Println(piscine.RockAndRoll(9))

fmt.Println(piscine.RockAndRoll(6))

}

And its output:

$ go run . | cat -e

rock$

$

roll$

$

rock and roll$

$

# **loafofbread**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

required

XP

Files to submit

Allowed functions

0.00 B

loafofbread.go

github.com/01-edu/z01.PrintRune, --allow-builtin

### **Instructions**

Write a function LoafOfBread() that takes a string and returns another one with words of 5 characters and skips the next character followed by newline \n.

* If there is a space in the middle of a string it should ignore it and get the next character until getting to a length of 5.
* If the string is less than 5 characters return "Invalid Output\n".

### **Expected function**

func LoafOfBread(str string) string {

}

### **Usage**

Here is a possible program to test your function:

package main

import (

"fmt"

"piscine"

)

func main() {

fmt.Print(piscine.LoafOfBread("deliciousbread"))

fmt.Print(piscine.LoafOfBread("This is a loaf of bread"))

fmt.Print(piscine.LoafOfBread("loaf"))

}

And its output:

$ go run . | cat -e

delic ousbr ad$

Thisi aloaf ofbre d$

Invalid Output$

Something is wrong ? [Submit an issue](https://github.com/01-edu/public/issues/new?body=loafofbread%0A---%0A&title=loafofbread+subject)

# **listpushback**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

required

XP

Files to submit

Allowed functions

0.00 B

listpushback.go

--allow-builtin

### **Instructions**

Write a function ListPushBack that inserts a new element NodeL at the end of the list l while using the structure List.

### **Expected function and structure**

type NodeL struct {

Data interface{}

Next \*NodeL

}

type List struct {

Head \*NodeL

Tail \*NodeL

}

func ListPushBack(l \*List, data interface{}) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

link := &piscine.List{}

piscine.ListPushBack(link, "Hello")

piscine.ListPushBack(link, "man")

piscine.ListPushBack(link, "how are you")

for link.Head != nil {

fmt.Println(link.Head.Data)

link.Head = link.Head.Next

}

}

And its output :

$ go run .

Hello

man

how are you

$

# **listpushfront**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

required

XP

Files to submit

Allowed functions

0.00 B

listpushfront.go

--allow-builtin

### **Instructions**

Write a function ListPushFront that inserts a new element NodeL at the beginning of the list l while using the structure List

### **Expected function and structure**

type NodeL struct {

Data interface{}

Next \*NodeL

}

type List struct {

Head \*NodeL

Tail \*NodeL

}

func ListPushFront(l \*List, data interface{}) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

link := &piscine.List{}

piscine.ListPushFront(link, "Hello")

piscine.ListPushFront(link, "man")

piscine.ListPushFront(link, "how are you")

it := link.Head

for it != nil {

fmt.Print(it.Data, " ")

it = it.Next

}

fmt.Println()

}

And its output :

$ go run .

how are you man Hello

$

# **listsize**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

required

XP

Files to submit

Allowed functions

0.00 B

listsize.go

--allow-builtin

### **Instructions**

Write a function ListSize that returns the number of elements in a linked list l.

### **Expected function and structure**

type NodeL struct {

Data interface{}

Next \*NodeL

}

type List struct {

Head \*NodeL

Tail \*NodeL

}

func ListSize(l \*List) int {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

link := &piscine.List{}

piscine.ListPushFront(link, "Hello")

piscine.ListPushFront(link, "2")

piscine.ListPushFront(link, "you")

piscine.ListPushFront(link, "man")

fmt.Println(piscine.ListSize(link))

}

And its output :

$ go run .

4

$

# **listlast**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

required

XP

Files to submit

Allowed functions

0.00 B

listlast.go

--allow-builtin

### **Instructions**

Write a function ListLast that returns the Data interface of the last element of a linked list l.

### **Expected function and structure**

type NodeL struct {

Data interface{}

Next \*NodeL

}

type List struct {

Head \*NodeL

Tail \*NodeL

}

func ListLast(l \*List) interface{} {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

link := &piscine.List{}

link2 := &piscine.List{}

piscine.ListPushBack(link, "three")

piscine.ListPushBack(link, 3)

piscine.ListPushBack(link, "1")

fmt.Println(piscine.ListLast(link))

fmt.Println(piscine.ListLast(link2))

}

And its output :

$ go run .

1

<nil>

$

Something is wrong ? [Su](https://github.com/01-edu/public/issues/new?body=listlast%0A---%0A&title=listlast+subject)

# **listclear**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

required

XP

Files to submit

Allowed functions

0.00 B

listclear.go

--allow-builtin

### **Instructions**

Write a function ListClear that deletes all nodes from a linked list l.

* Tip: assign the list's pointer to nil.

### **Expected function and structure**

func ListClear(l \*List) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

type List = piscine.List

type Node = piscine.NodeL

func PrintList(l \*List) {

link := l.Head

for link != nil {

fmt.Print(link.Data, " -> ")

link = link.Next

}

fmt.Println(nil)

}

func main() {

link := &List{}

piscine.ListPushBack(link, "I")

piscine.ListPushBack(link, 1)

piscine.ListPushBack(link, "something")

piscine.ListPushBack(link, 2)

fmt.Println("------list------")

PrintList(link)

piscine.ListClear(link)

fmt.Println("------updated list------")

PrintList(link)

}

And its output :

$ go run .

------list------

I -> 1 -> something -> 2 -> <nil>

------updated list------

<nil>

$

# **listat**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

required

XP

Files to submit

Allowed functions

0.00 B

listat.go

--allow-builtin

### **Instructions**

Write a function ListAt that takes a pointer to the head of the list l and an int pos as parameters. This function should return the pointer to the NodeL in the position pos of the linked list l.

* In case of error the function should return nil.

### **Expected function and structure**

type NodeL struct {

Data interface{}

Next \*NodeL

}

func ListAt(l \*NodeL, pos int) \*NodeL{

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

link := &piscine.List{}

piscine.ListPushBack(link, "hello")

piscine.ListPushBack(link, "how are")

piscine.ListPushBack(link, "you")

piscine.ListPushBack(link, 1)

fmt.Println(piscine.ListAt(link.Head, 3).Data)

fmt.Println(piscine.ListAt(link.Head, 1).Data)

fmt.Println(piscine.ListAt(link.Head, 7))

}

And its output :

$ go run .

1

how are

<nil>

$

# **listreverse**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

required

XP

Files to submit

Allowed functions

0.00 B

listreverse.go

--allow-builtin

### **Instructions**

Write a function ListReverse that reverses the order of the elements of a given linked list l.

### **Expected function and structure**

type NodeL struct {

Data interface{}

Next \*NodeL

}

type List struct {

Head \*NodeL

Tail \*NodeL

}

func ListReverse(l \*List) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

link := &piscine.List{}

piscine.ListPushBack(link, 1)

piscine.ListPushBack(link, 2)

piscine.ListPushBack(link, 3)

piscine.ListPushBack(link, 4)

piscine.ListReverse(link)

it := link.Head

for it != nil {

fmt.Println(it.Data)

it = it.Next

}

fmt.Println("Tail", link.Tail)

fmt.Println("Head", link.Head)

}

And its output :

$ go run .

4

3

2

1

Tail &{1 <nil>}

Head &{4 0xc42000a140}

$

# **listforeach**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

required

XP

Files to submit

Allowed functions

0.00 B

listforeach.go

--allow-builtin

### **Instructions**

Write a function ListForEach that applies a function given as argument to the data within each node of the list l.

* The function given as argument must have a pointer as argument: l \*List
* Copy the functions Add2\_node and Subtract3\_node in the same file as the function ListForEach is defined.

### **Expected function and structure**

type NodeL struct {

Data interface{}

Next \*NodeL

}

type List struct {

Head \*NodeL

Tail \*NodeL

}

func ListForEach(l \*List, f func(\*NodeL)) {

}

func Add2\_node(node \*NodeL) {

switch node.Data.(type) {

case int:

node.Data = node.Data.(int) + 2

case string:

node.Data = node.Data.(string) + "2"

}

}

func Subtract3\_node(node \*NodeL) {

switch node.Data.(type) {

case int:

node.Data = node.Data.(int) - 3

case string:

node.Data = node.Data.(string) + "-3"

}

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

link := &piscine.List{}

piscine.ListPushBack(link, "1")

piscine.ListPushBack(link, "2")

piscine.ListPushBack(link, "3")

piscine.ListPushBack(link, "5")

piscine.ListForEach(link, piscine.Add2\_node)

it := link.Head

for it != nil {

fmt.Println(it.Data)

it = it.Next

}

}

And its output :

$ go run .

12

22

32

52

$

# **listforeachif**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

required

XP

Files to submit

Allowed functions

0.00 B

listforeachif.go

--allow-builtin

### **Instructions**

Write a function ListForEachIf that applies a function given as argument to the data within some of the nodes of the list l.

* This function receives two functions:
  + f is a function that is applied to the node.
  + cond is a function that returns a boolean and it will be used to determine if the function f should be applied to the node.
* The function given as argument must have a pointer \*NodeL as argument.

### **Expected function and structure**

type NodeL struct {

Data interface{}

Next \*NodeL

}

type List struct {

Head \*NodeL

Tail \*NodeL

}

func IsPositiveNode(node \*NodeL) bool {

switch node.Data.(type) {

case int, float32, float64, byte:

return node.Data.(int) > 0

default:

return false

}

}

func IsAlNode(node \*NodeL) bool {

switch node.Data.(type) {

case int, float32, float64, byte:

return false

default:

return true

}

}

func ListForEachIf(l \*List, f func(\*NodeL), cond func(\*NodeL) bool) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"piscine"

"fmt"

)

func PrintElem(node \*piscine.NodeL) {

fmt.Println(node.Data)

}

func StringToInt(node \*piscine.NodeL) {

node.Data = 2

}

func PrintList(l \*piscine.List) {

it := l.Head

for it != nil {

fmt.Print(it.Data, "->")

it = it.Next

}

fmt.Print("nil","\n")

}

func main() {

link := &piscine.List{}

piscine.ListPushBack(link, 1)

piscine.ListPushBack(link, "hello")

piscine.ListPushBack(link, 3)

piscine.ListPushBack(link, "there")

piscine.ListPushBack(link, 23)

piscine.ListPushBack(link, "!")

piscine.ListPushBack(link, 54)

PrintList(link)

fmt.Println("--------function applied--------")

piscine.ListForEachIf(link, PrintElem, piscine.IsPositiveNode)

piscine.ListForEachIf(link, StringToInt, piscine.IsAlNode)

fmt.Println("--------function applied--------")

PrintList(link)

fmt.Println()

}

And its output :

$ go run .

1->hello->3->there->23->!->54->nil

--------function applied--------

1

3

23

54

--------function applied--------

1->2->3->2->23->2->54->nil

$

# **listfind**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

optional

XP

Files to submit

Allowed functions

0.00 B

listfind.go

--allow-builtin

### **Instructions**

Write a function ListFind that returns the address of the data interface of the first node in the list l that is determined to be equal to ref by the function CompStr.

* For this exercise the function CompStr must be used.

### **Expected function and structure**

type NodeL struct {

Data interface{}

Next \*NodeL

}

type List struct {

Head \*NodeL

Tail \*NodeL

}

func CompStr(a, b interface{}) bool {

return a == b

}

func ListFind(l \*List, ref interface{}, comp func(a, b interface{}) bool) \*interface{} {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func main() {

link := &piscine.List{}

piscine.ListPushBack(link, "hello")

piscine.ListPushBack(link, "hello1")

piscine.ListPushBack(link, "hello2")

piscine.ListPushBack(link, "hello3")

found := piscine.ListFind(link, interface{}("hello2"), piscine.CompStr)

fmt.Println(found)

fmt.Println(\*found)

}

And its output :

$ go run .

0xc42000a0a0

hello2

$

### **Note**

* The address may be different in each execution of the program.

# **listremoveif**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 25

optional

XP

Files to submit

Allowed functions

0.00 B

listremoveif.go

--allow-builtin

### **Instructions**

Write a function ListRemoveIf that removes all elements that are equal to the data\_ref in the argument of the function.

### **Expected function and structure**

type NodeL struct {

Data interface{}

Next \*NodeL

}

type List struct {

Head \*NodeL

Tail \*NodeL

}

func ListRemoveIf(l \*List, data\_ref interface{}) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func PrintList(l \*piscine.List) {

it := l.Head

for it != nil {

fmt.Print(it.Data, " -> ")

it = it.Next

}

fmt.Print(nil, "\n")

}

func main() {

link := &piscine.List{}

link2 := &piscine.List{}

fmt.Println("----normal state----")

piscine.ListPushBack(link2, 1)

PrintList(link2)

piscine.ListRemoveIf(link2, 1)

fmt.Println("------answer-----")

PrintList(link2)

fmt.Println()

fmt.Println("----normal state----")

piscine.ListPushBack(link, 1)

piscine.ListPushBack(link, "Hello")

piscine.ListPushBack(link, 1)

piscine.ListPushBack(link, "There")

piscine.ListPushBack(link, 1)

piscine.ListPushBack(link, 1)

piscine.ListPushBack(link, "How")

piscine.ListPushBack(link, 1)

piscine.ListPushBack(link, "are")

piscine.ListPushBack(link, "you")

piscine.ListPushBack(link, 1)

PrintList(link)

piscine.ListRemoveIf(link, 1)

fmt.Println("------answer-----")

PrintList(link)

}

And its output :

$ go run .

----normal state----

1 -> <nil>

------answer-----

<nil>

----normal state----

1 -> Hello -> 1 -> There -> 1 -> 1 -> How -> 1 -> are -> you -> 1 -> <nil>

------answer-----

Hello -> There -> How -> are -> you -> <nil>

$

# **listmerge**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 26

optional

XP

Files to submit

Allowed functions

0.00 B

listmerge.go

--allow-builtin

### **Instructions**

Write a function ListMerge that places elements of a list l2 at the end of another list l1.

* New elements should not be created!

### **Expected function and structure**

type NodeL struct {

Data interface{}

Next \*NodeL

}

type List struct {

Head \*NodeL

Tail \*NodeL

}

func ListMerge(l1 \*List, l2 \*List) {

}

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func PrintList(l \*piscine.List) {

it := l.Head

for it != nil {

fmt.Print(it.Data, " -> ")

it = it.Next

}

fmt.Print(nil, "\n")

}

func main() {

link := &piscine.List{}

link2 := &piscine.List{}

piscine.ListPushBack(link, "a")

piscine.ListPushBack(link, "b")

piscine.ListPushBack(link, "c")

piscine.ListPushBack(link, "d")

fmt.Println("-----first List------")

PrintList(link)

piscine.ListPushBack(link2, "e")

piscine.ListPushBack(link2, "f")

piscine.ListPushBack(link2, "g")

piscine.ListPushBack(link2, "h")

fmt.Println("-----second List------")

PrintList(link2)

fmt.Println("-----Merged List-----")

piscine.ListMerge(link, link2)

PrintList(link)

}

And its output :

$ go run .

-----first List------

a -> b -> c -> d -> <nil>

-----second List------

e -> f -> g -> h -> <nil>

-----Merged List-----

a -> b -> c -> d -> e -> f -> g -> h -> <nil>

$

# **listsort**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 26

optional

XP

Files to submit

Allowed functions

0.00 B

listsort.go

--allow-builtin

### **Instructions**

Write a function ListSort that sorts the nodes of a linked list by ascending order.

### **Expected function and structure**

type NodeI struct {

Data int

Next \*NodeI

}

func ListSort(l \*NodeI) \*NodeI {

}

You will use **this** NodeI structure in subsequent exercises.

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func PrintList(l \*piscine.NodeI) {

it := l

for it != nil {

fmt.Print(it.Data, " -> ")

it = it.Next

}

fmt.Print(nil, "\n")

}

func listPushBack(l \*piscine.NodeI, data int) \*piscine.NodeI {

n := &piscine.NodeI{Data: data}

if l == nil {

return n

}

iterator := l

for iterator.Next != nil {

iterator = iterator.Next

}

iterator.Next = n

return l

}

func main() {

var link \*piscine.NodeI

link = listPushBack(link, 5)

link = listPushBack(link, 4)

link = listPushBack(link, 3)

link = listPushBack(link, 2)

link = listPushBack(link, 1)

PrintList(piscine.ListSort(link))

}

And its output :

$ go run .

1 -> 2 -> 3 -> 4 -> 5 -> <nil>

$

# **sortlistinsert**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 26

optional

XP

Files to submit

Allowed functions

0.00 B

sortlistinsert.go

--allow-builtin

### **Instructions**

Create a function named SortListInsert, which accepts a pointer to the head of a sorted linked list and an integer.

The function should insert a new element into the list, with its Data set to the value of the integer. The element should be inserted so that the linked list remains sorted in ascending order by Data. The function should return the head of the list.

Your function will only be tested with sorted linked lists.

### **Expected function and structure**

func SortListInsert(l \*NodeI, data\_ref int) \*NodeI{

}

You have already defined the NodeI structure in the listsort exercise.

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func PrintList(l \*piscine.NodeI) {

it := l

for it != nil {

fmt.Print(it.Data, " -> ")

it = it.Next

}

fmt.Print(nil, "\n")

}

func listPushBack(l \*piscine.NodeI, data int) \*piscine.NodeI {

n := &piscine.NodeI{Data: data}

if l == nil {

return n

}

iterator := l

for iterator.Next != nil {

iterator = iterator.Next

}

iterator.Next = n

return l

}

func main() {

var link \*piscine.NodeI

link = listPushBack(link, 1)

link = listPushBack(link, 4)

link = listPushBack(link, 9)

PrintList(link)

link = piscine.SortListInsert(link, -2)

link = piscine.SortListInsert(link, 2)

PrintList(link)

}

And its output :

$ go run .

1 -> 4 -> 9 -> <nil>

-2 -> 1 -> 2 -> 4 -> 9 -> <nil>

$

# **sortedlistmerge**

succeeded

This exercise is not in the time scope anymore so it won't reward you any XP.

Level 26

optional

XP

Files to submit

Allowed functions

0.00 B

sortedlistmerge.go

--allow-builtin

### **Instructions**

Write a function SortedListMerge that merges two lists n1 and n2 in ascending order.

* During the tests n1 and n2 will already be initially sorted.

### **Expected function and structure**

func SortedListMerge(n1 \*NodeI, n2 \*NodeI) \*NodeI {

}

You have already defined the NodeI structure in the listsort exercise.

### **Usage**

Here is a possible program to test your function :

package main

import (

"fmt"

"piscine"

)

func PrintList(l \*piscine.NodeI) {

it := l

for it != nil {

fmt.Print(it.Data, " -> ")

it = it.Next

}

fmt.Print(nil, "\n")

}

func listPushBack(l \*piscine.NodeI, data int) \*piscine.NodeI {

n := &piscine.NodeI{Data: data}

if l == nil {

return n

}

iterator := l

for iterator.Next != nil {

iterator = iterator.Next

}

iterator.Next = n

return l

}

func main() {

var link \*piscine.NodeI

var link2 \*piscine.NodeI

link = listPushBack(link, 3)

link = listPushBack(link, 5)

link = listPushBack(link, 7)

link2 = listPushBack(link2, -2)

link2 = listPushBack(link2, 9)

PrintList(piscine.SortedListMerge(link2, link))

}

And its output :

$ go run .

-2 -> 3 -> 5 -> 7 -> 9 -> <nil>

$

Something is wrong