01-edu Practice Questions

Checkpoint 01 - Questions Only

⚠️ **EXAM RULES REMINDER**

- **Level 0:**

  Must be `package main`.
- **Level 1-3:**

  Must be `package piscine`.
- **Forbidden:**

  Do not use `fmt` or `strings` unless specified.
- **Allowed:**

  `github.com/01-edu/z01` is usually allowed for printing.

Level 0: Basic Printing Programs

Create a standalone program (`main.go`) for each task.

1. only1 / onlya / onlyz

Instructions: Write a program that displays a specific character (e.g., '1', 'a', or 'z') to the standard output, followed by a newline.

```
// Expected Structure
package main
import "github.com/01-edu/z01"

func main() {
    // TODO: Print the character and a newline
}
```

2. hello

Instructions: Write a program that displays "Hello World!" to the standard output, followed by a newline. You must loop through the string.

```go
// Expected Structure
package main
import "github.com/01-edu/z01"

func main() {
    // TODO: Print "Hello World!" using z01
}
```

3. displayalpham

Instructions: Write a program that displays the alphabet in reverse order (from 'z' to 'a') to the standard output, followed by a newline.

```go
// Expected Structure
package main
import "github.com/01-edu/z01"

func main() {
    // TODO: Print z...a
}
```

Level 1: Logic & Validation

Write the function inside a file. Use `package piscine`.

4. checknumber

Instructions: Write a function that returns `true` if the string contains **only** digits, and `false` otherwise. Empty strings return false.

```go
package piscine

func CheckNumber(arg string) bool {
    // TODO
}
```

5. countalpha

Instructions: Write a function that returns the number of alphabetic characters (letters a-z, A-Z) in a string.

```go
package piscine

func CountAlpha(s string) int {
    // TODO
}
```

6. countcharacter

Instructions: Write a function that returns the number of times the character `c` appears in the string `s`.

```go
package piscine

func CountCharacter(s string, c rune) int {
    // TODO
}
```

7. printif

Instructions: Write a function that returns `"G\n"` if the string length is **3 or more** (or empty). Otherwise, return `"Invalid Input\n"`.

```go
package piscine

func PrintIf(str string) string {
    // TODO
}
```

8. printifnot

Instructions: Write a function that returns `"G\n"` if the string length is **less than 3**. Otherwise, return `"Invalid Input\n"`.

```go
package piscine

func PrintIfNot(str string) string {
    // TODO
}
```

9. rectperimeter

Instructions: Write a function that returns the perimeter of a rectangle. If either argument is negative, return `-1`.

*// Formula: 2 * (w + h)*

```go
package piscine

func RectPerimeter(w, h int) int {
    // TODO
}
```

10. retainfirsthalf

Instructions: Write a function that returns the first half of the string. If length is 1, return the string. If empty, return empty.

```go
package piscine

func RetainFirstHalf(str string) string {
    // TODO
}
```

Level 2/3: Algorithms

Intermediate difficulty. Use `package piscine`.

11. digitlen

Instructions: Write a function that returns the number of digits in `n` using the specified `base`.

- If `base` < 2 or > 36, return `-1`.
- Handle negative numbers.
- Handle 0 (length is 1).

```go
package piscine

func DigitLen(n, base int) int {
    // TODO
}
```

12. fishandchips

Instructions: Return a string based on the number `n`:

- Divisible by 2 AND 3: `"fish and chips"`
- Divisible by 2: `"fish"`
- Divisible by 3: `"chips"`
- Negative: `"error: number is negative"`
- None of the above: `"error: non divisible"`

```go
package piscine

func FishAndChips(n int) string {
    // TODO
}
```

13. firstword

Instructions: Write a function that returns the first word of a string. The returned string must end with a newline. Ignore leading spaces.

```go
package piscine

func FirstWord(s string) string {
    // TODO
}
```

14. lastword

Instructions: Write a function that returns the last word of a string. The returned string must end with a newline. Ignore trailing spaces.

```go
package piscine

func LastWord(s string) string {
    // TODO
}
```