01-edu Checkpoint 01

Comprehensive Study Guide for Levels 0-3

⚠️ **IMPORTANT EXAM RULES**

- **Package Names:**

  Use `package main` for standalone programs (Level 0). Use `package piscine` for functions (Level 1-3).
- **Imports:**

  Do NOT import "fmt" or "strings" unless explicitly allowed. Use `"github.com/01-edu/z01"` for printing.
- **Formatting:**

  Always check your syntax. Precise output matters.

Level 0: Basic Printing Programs

These must use `package main` and `z01.PrintRune`.

1. only1 / onlya / onlyz

```
package main

import "github.com/01-edu/z01"

func main() {
    // Change '1' to 'a', 'z', 'f' based on the question
    z01.PrintRune('1')
    z01.PrintRune('\n')
}
```

2. hello (Hello World)

```go
package main

import "github.com/01-edu/z01"

func main() {
    str := "Hello World!"
    for _, r := range str {
        z01.PrintRune(r)
    }
    z01.PrintRune('\n')
}
```

3. displayalpham (Reverse Alphabet)

```go
package main

import "github.com/01-edu/z01"

func main() {
    for i := 'z'; i >= 'a'; i-- {
        z01.PrintRune(i)
    }
    z01.PrintRune('\n')
}
```

Level 1: Logic & Validation

These are functions. Use `package piscine`. Do not use main.

4. checknumber

// Returns true if string contains ONLY digits.

```go
package piscine

func CheckNumber(arg string) bool {
    if arg == "" {
        return false
    }
    for _, r := range arg {
        if r < '0' || r > '9' {
            return false
        }
    }
    return true
}
```

5. countalpha

// Returns count of alphabetic characters (a-z, A-Z).

```go
package piscine

func CountAlpha(s string) int {
    count := 0
    for _, r := range s {
        if (r >= 'a' && r <= 'z') || (r >= 'A' && r <= 'Z') {
            count++
        }
    }
    return count
}
```

6. countcharacter

// Returns number of times character 'c' appears.

```go
package piscine

func CountCharacter(s string, c rune) int {
    count := 0
    for _, r := range s {
        if r == c {
            count++
        }
    }
    return count
}
```

7. printif

// Returns "G\n" if len >= 3, else "Invalid Input\n".

```go
package piscine

func PrintIf(str string) string {
    if len(str) >= 3 || str == "" {
        return "G\n"
    }
    return "Invalid Input\n"
}
```

8. printifnot

// Returns "G\n" if len < 3, else "Invalid Input\n".

```go
package piscine

func PrintIfNot(str string) string {
    if len(str) < 3 {
        return "G\n"
    }
    return "Invalid Input\n"
}
```

9. rectperimeter

// Returns perimeter. Returns -1 if negative inputs.

```go
package piscine

func RectPerimeter(w, h int) int {
    if w < 0 || h < 0 {
        return -1
    }
    return 2 * (w + h)
}
```

10. retainfirsthalf

// Returns first half of string. Handles length 1 edge case.

```go
package piscine

func RetainFirstHalf(str string) string {
    if len(str) == 1 {
        return str
    }
    half := len(str) / 2
    return str[:half]
}
```

Level 2/3: Algorithms

Intermediate logic involving math or advanced string parsing.

11. digitlen

// Counts digits of n in a specific base.

```go
package piscine

func DigitLen(n, base int) int {
    if base < 2 || base > 36 {
        return -1
    }
    if n < 0 {
        n = -n
    }
    if n == 0 {
        return 1
    }
    count := 0
    for n != 0 {
        n = n / base
        count++
    }
    return count
}
```

12. fishandchips

// FizzBuzz style logic. Order of checks is critical.

```go
package piscine

func FishAndChips(n int) string {
    if n < 0 {
        return "error: number is negative"
    }
    // Must check BOTH first
    if n%2 == 0 && n%3 == 0 {
        return "fish and chips"
    }
    if n%2 == 0 {
        return "fish"
    }
    if n%3 == 0 {
        return "chips"
    }
    return "error: non divisible"
}
```

13. firstword

// Returns first word, ignores leading spaces.

```go
package piscine

func FirstWord(s string) string {
    word := ""
    found := false
    for _, r := range s {
        if r != ' ' {
            found = true
            word += string(r)
        } else if found {
            break
        }
    }
    return word + "\n"
}
```

14. lastword

// Returns last word, ignores trailing spaces.

```
package piscine

func LastWord(s string) string {
    end := len(s) - 1
    // Skip trailing spaces from the back
    for end >= 0 && s[end] == ' ' {
        end--
    }

    // Find start of the word
    start := end
    for start >= 0 && s[start] != ' ' {
        start--
    }

    if end < 0 {
        return "\n"
    }

    // Use +1 because start is at the space BEFORE the word
    return s[start+1 : end+1] + "\n"
}
```

eof