Задача 1: Исправление работы с указателями

Условие:
Исправить функцию `printNumber`, чтобы она работала корректно. Сигнатуру функции менять нельзя.

```
func printNumber(ptrToNumber interface{}) {
        if ptrToNumber != nil {
                fmt.Println(*ptrToNumber.(*int))
        } else {
                fmt.Println("nil")
        }
}

func main() {
        v := 10
        printNumber(&v)
        var pv *int
        printNumber(pv)
        pv = &v
        printNumber(pv)
}
```

Требуется:

- Исправить возможные паники и ошибки приведения типов.
- Обработать ситуацию с `nil` указателем.

Задача 2: Работа с интерфейсами и нулевыми значениями

Условие:
Что выведет код и почему?

```
package main

import "fmt"

type MyError struct{}

func (MyError) Error() string {
    return "MyError!"
}
```

```
func errorHandler(err error) {
    if err != nil {
        fmt.Println("Error:", err)
    }
}

func main() {
    var err *MyError
    errorHandler(err)
    err = &MyError{}
    errorHandler(err)
}
```

Требуется:

- Объяснить, почему в первом вызове ошибки не выводится сообщение.
- Что происходит при передаче nil-значения интерфейса.

Задача 3: Поведение append

Условие:

```
var foo []int
var bar []int

foo = append(foo, 1)
foo = append(foo, 2)
foo = append(foo, 3)
bar = append(foo, 4)
foo = append(foo, 5)

fmt.Println(foo, bar)
```

Требуется:

- Предсказать результат вывода.
- Разобраться с механизмом выделения памяти для слайсов.

Задача 4: Работа с мапами

Условие:

```
func main() {
    var m map[string]int
    for _, word := range []string{"hello", "world", "from", "the", "best", "language",
"in", "the", "world"} {
        m[word]++
    }
    for k, v := range m {
        fmt.Println(k, v)
    }
}
```

Требуется:

- Объяснить, что произойдет при попытке использования неинициализированной мапы.
- Разобрать, как можно избежать паники в подобных ситуациях.

Задача 5: Изменение строки

Условие:

```
package main

import "fmt"

func main() {
    str := "Привет"
    str[2] = 'e'
    fmt.Println(str)
}
```

Требуется:

- Объяснить, почему код вызывает ошибку компиляции.
- Как можно изменить символ в строке корректно?

Задача 6: Генератор паролей

Условие:

1. Написать функцию генератор паролей, которая принимает целое число `n`, а на выходе строка длины `n` из символов `a-zA-Z0-9`.
2. Можно ли угадать генерируемую строку? Как тестировать такую функцию?

Задача 7: Использование каналов и WaitGroup

Условие:

Что выведет следующий код? Исправьте все проблемы.

```
func main() {
    ch := make(chan int)
    wg := &sync.WaitGroup{}
    wg.Add(3)
    for i := 0; i < 3; i++ {
        go func(v int) {
            defer wg.Done()
            ch <- v * v
        }(i)
    }
    wg.Wait()
    close(ch)
    var sum int
    for v := range ch {
        sum += v
    }
    fmt.Printf("result: %d\n", sum)
}
```

Требуется:

- Найти ошибки и предложить исправленный вариант.
- Объяснить, почему возникают deadlock и гонки данных.

Задача 8: Работа с каналами

Условие:

```
func main() {
    ch := make(chan bool)
    ch <- true
    go func() {
        <-ch
    }()
    ch <- true
}
```

Требуется:

- Найти проблемы и исправить код.

Задача 9: Merge n каналов

Условие:

Написать функцию, которая объединяет несколько входных каналов в один выходной.

```
func merge(channels ...chan int) chan int {
    // Реализация
}
```

Требуется:

- Реализовать объединение каналов.
- Закрыть все каналы при закрытии одного из входных.