← Учебный лагерь APG1. День07

Отзы

Отправить проект

Завершить проект

Лагерь выживания

Показать все

Учебный лагерь APG1. День00	100 %
Учебный лагерь APG1. День01	100 %
Учебный лагерь APG1. День02	100 %
Учебный лагерь APG1. День03	0 %
Учебный лагерь APG1. День04	100 %
Учебный лагерь APG1. Команда00	0 %
Учебный лагерь APG1. День05	
Учебный лагерь APG1. День06	
Учебный лагерь APG1. День07	
Учебный лагерь APG1. День08	
Учебный лагерь APG1. День09	
Учебный лагерь APG1. Команда01	

Частный проект Git

ssh://git@repos-ssh.21-school.ru:2289/students/Go_Day07.ID_376229/mlarra_student.21_scho...

Копировать ссылку Открыть

Задача

День 07. Учебный лагерь

Сумка денег

Содержание

Глава I

- Ваши программы не должны завершаться неожиданно (выдавая ошибку при правильном вводе). Если это произойдет, ваш проект будет считаться неработоспособным и получит 0 баллов при оценке.
- Мы рекомендуем вам создать тестовые программы для вашего проекта, даже если эту работу не нужно будет отправлять и она не будет оцениваться. Это даст вам возможность легко проверить свою работу и работу ваших коллег. Вы найдете эти тесты особенно полезными во время вашей защиты. Действительно, во время защиты вы можете использовать свои тесты и/или тесты коллеги, которого вы оцениваете.
- Отправьте свою работу в назначенный репозиторий git. Оцениваться будет только работа в репозитории git.
- Если ваш код использует внешние зависимости, он должен использовать модули Go для управления ими.

Глава II

Правила дня

- Вы должны только сдать `*.go` файлы и (в случае внешних зависимостей) `go.mod` + `go.sum`
- Ваш код для этой задачи должен быть собран с помощью всего лишь `go build`
- Все ваши тесты должны быть запущены путем вызова стандартного `go test ./...`

Глава III

вступление

«Есть несколько областей, где мы считаем надежность и скорость критически важными. Области, которые напрямую влияют на жизнь людей — медицина, безопасность самолетов, финансы. Конечно, это означает, что мы тщательно просматриваем каждую деталь нашего продукта, прежде чем выпускать его для широкой публики. Дамы и господа, представляю вам... Денежного мешка!"

Глава IV

Упражнение 00: Королевская награда

Вы продолжаете слушать голос генерального директора, но ваши глаза смотрят на код на вашем ноутбуке.

Иногда кажется, что люди всегда будут использовать монеты для оплаты вещей. В прачечных, торговых автоматах или музыкальных шкатулках по-прежнему принято принимать в качестве оплаты только куски металла. Но люди иногда ненавидят стоять в очередях и ждать, пока кто-то другой соберет монеты и бросит их одну за другой. Почему люди не могут просто всегда использовать минимальное количество монет, чтобы не замедлять всех остальных?

Это довольно известная проблема, и ваш коллега уже написал код и загрузил его вам для обзора:

```
func minCoins(val int, coins []int) []int {
    res := make([]int, 0)
    i := len(coins) - 1
    for i >= 0 {
        for val >= coins[i] {
            val -= coins[i]
            res = append(res, coins[i])
        }
        i -= 1
    }
    return res
}
```

Он принимает необходимое количество и отсортированный срез уникальных номиналов монет. Это может быть что-то вроде [1,5,10,50,100,500,1000] или что-то экзотическое, например [1,3,4,7,13,15]. Предполагается, что на выходе должен быть фрагмент монет минимального размера, который можно использовать для выражения значения (например, для 13 и [1,5,10] это должно дать вам [10,1,1,1]).

Проблема в том, что вы нутром чувствуете, что с этим кодом что-то не так. Ваша цель — написать несколько тестов (в `*_test.go` файлах) для этого кода, которые покажут, что он дает неверные результаты. Кроме того, вам нужно написать отдельную функцию (вы должны назвать ее `minCoins2`), которая будет иметь те же параметры, но будет успешно обрабатывать эти случаи. В случае, если в срезе номиналов присутствуют дубликаты или он не отсортирован, функция все равно должна выдавать правильный результат. Если он пуст, должен быть возвращен пустой фрагмент.

Глава V

Упражнение 01: Жажда скорости

Теперь, когда у вас есть новая версия кода из EX00, давайте проверим ее на производительность. Ваши цели здесь:

- get a list of top 10 functions in your code (calling your function with some test data) that your CPU spends the most time executing (you should use Go's builtin tools for that). Submit that list as file `top10.txt`
- write a benchmark version of your tests that will compare the performance of your new code vs. the old one, especially while using relatively big denomination slice. If you find any more optimizations during this phase, feel free to submit newer version of your `minCoins2` function calling it `minCoins2Optimized` (not a required step)

Chapter VI

Exercise 02: Elder Scrolls

Теперь, когда вы исправили ошибку и написали несколько тестов для своего кода, пришло время создать для него некоторую документацию. Опишите в комментариях к вашему коду, чем ваше решение отличается от решения по умолчанию и какие оптимизации вы использовали. Затем используйте любой инструмент, который вам удастся найти, для создания HTML-документации на основе этих комментариев.

Указания о том, как воспроизвести генерацию документации, также должны быть включены в комментарии. Сохранение HTML-страниц из веб-браузера считается мошенничеством (хотя это и не запрещено строго, поэтому, если вы не можете сделать это другим способом, просто напишите об этом в комментариях).

Отправьте сгенерированную документацию (HTML-файлы + статические изображения, js и css), упакованные в `docs.zip` архив.