Отчет о прохождении 3 этапа внешних курсов

Продвинутые темы

Артемов Данил Сергеевич НБИбд-01-24

Содержание

# 1 Цель работы

Ознакомиться с функционалом операционной системы Linux.

# 2 Задание

Просмотреть видео и на основе полученной информации пройти тестовые задания.

# 3 Теоретическое введение

Линукс - в части случаев GNU/Linux — семейство Unix-подобных операционных систем на базе ядра Linux, включающих тот или иной набор утилит и программ проекта GNU, и, возможно, другие компоненты. Как и ядро Linux, системы на его основе, как правило, создаются и распространяются в соответствии с моделью разработки свободного и открытого программного обеспечения.

# 4 Выполнение лабораторной работы

3 Этап: (рис. [1](#fig:001), [2](#fig:002), [3](#fig:003), [4](#fig:004), [5](#fig:005), [6](#fig:006), [7](#fig:007), [8](#fig:008), [9](#fig:009), [10](#fig:010), [11](#fig:011), [12](#fig:012), [13](#fig:013), [14](#fig:014), [15](#fig:015), [16](#fig:016), [17](#fig:017), [18](#fig:018), [19](#fig:019), [20](#fig:020), [21](#fig:021), [22](#fig:022), [23](#fig:023), [24](#fig:024), [25](#fig:025), [26](#fig:026), [27](#fig:027), [28](#fig:028), [29](#fig:029), [30](#fig:030), [31](#fig:031), [32](#fig:032), [33](#fig:033), [34](#fig:034), [35](#fig:035), [36](#fig:036), [37](#fig:037), [38](#fig:038), [39](#fig:039), [40](#fig:040), [41](#fig:041)).

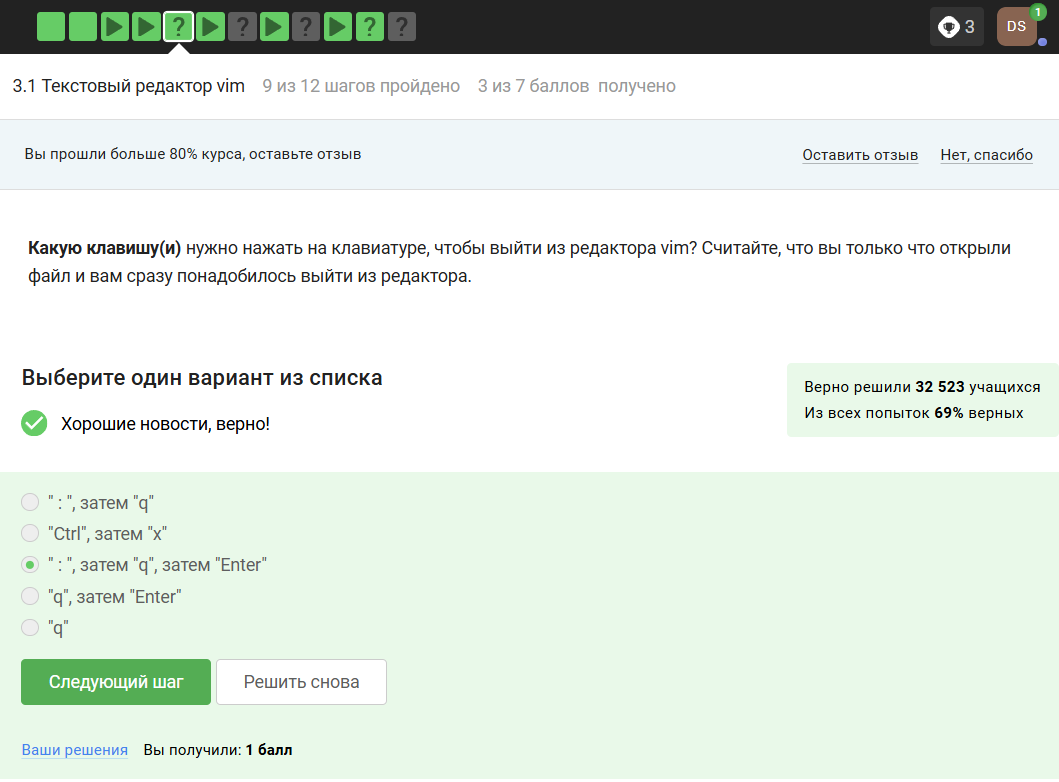


Figure 1: Задание 1

Стоит упомянуть, что у редактора vim есть туториал, который позволяет разобраться с командами, необходимыми для стандартной работы. За выход из редактора отвечают следующие команды:

* ZQ - выйти без сохранения
* :q! - выйти без сохранения
* ZZ - записать файл и выйти (если файл не изменяли, то записываться он не будет)
* :wq - записать файл и выйти
* :x - записать файл и выйти
* :w - записать файл
* :sav filename - “сохранить как”
* :w filename - “сохранить как”
* :w! - записать файл

Как мы видим, вариантов много, при этом каждый сможет найти тот, который подойдёт под конкретную ситуацию.

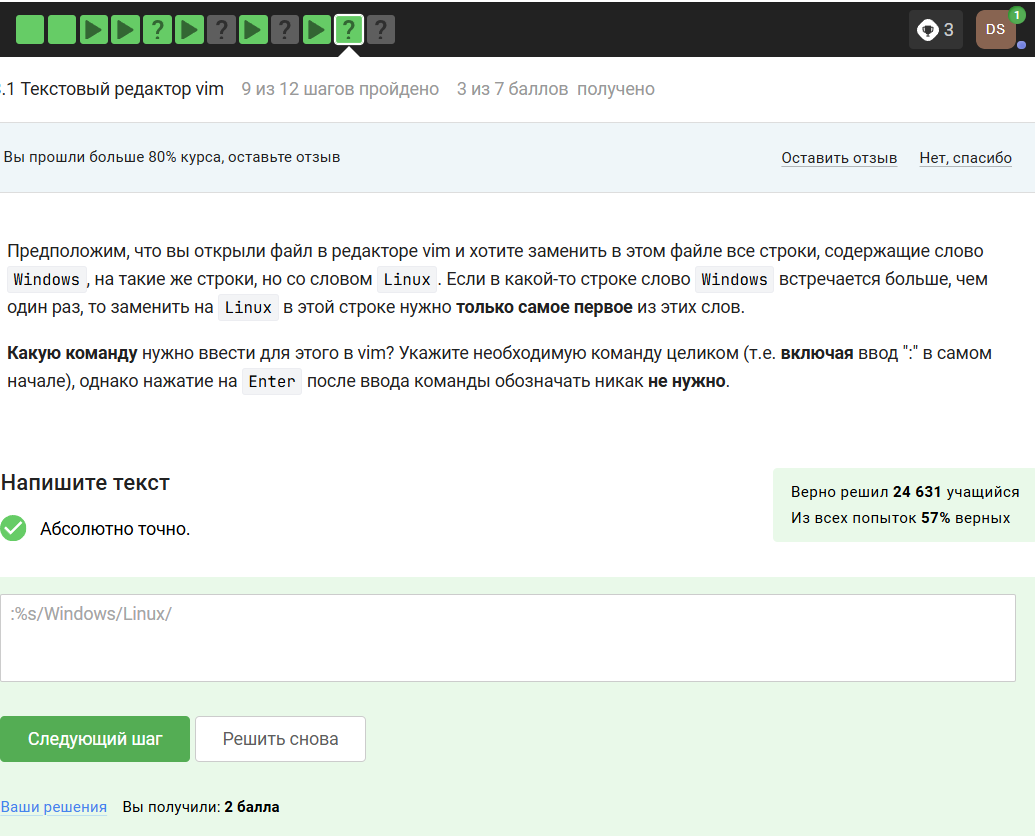


Figure 4: Задание 4

Поиск и замена в редакторе работают по следующей схеме:

:{пределы}s/{что заменяем}/{на что заменяем}/{опции}

Для замены во всем файле можно использовать символ %.

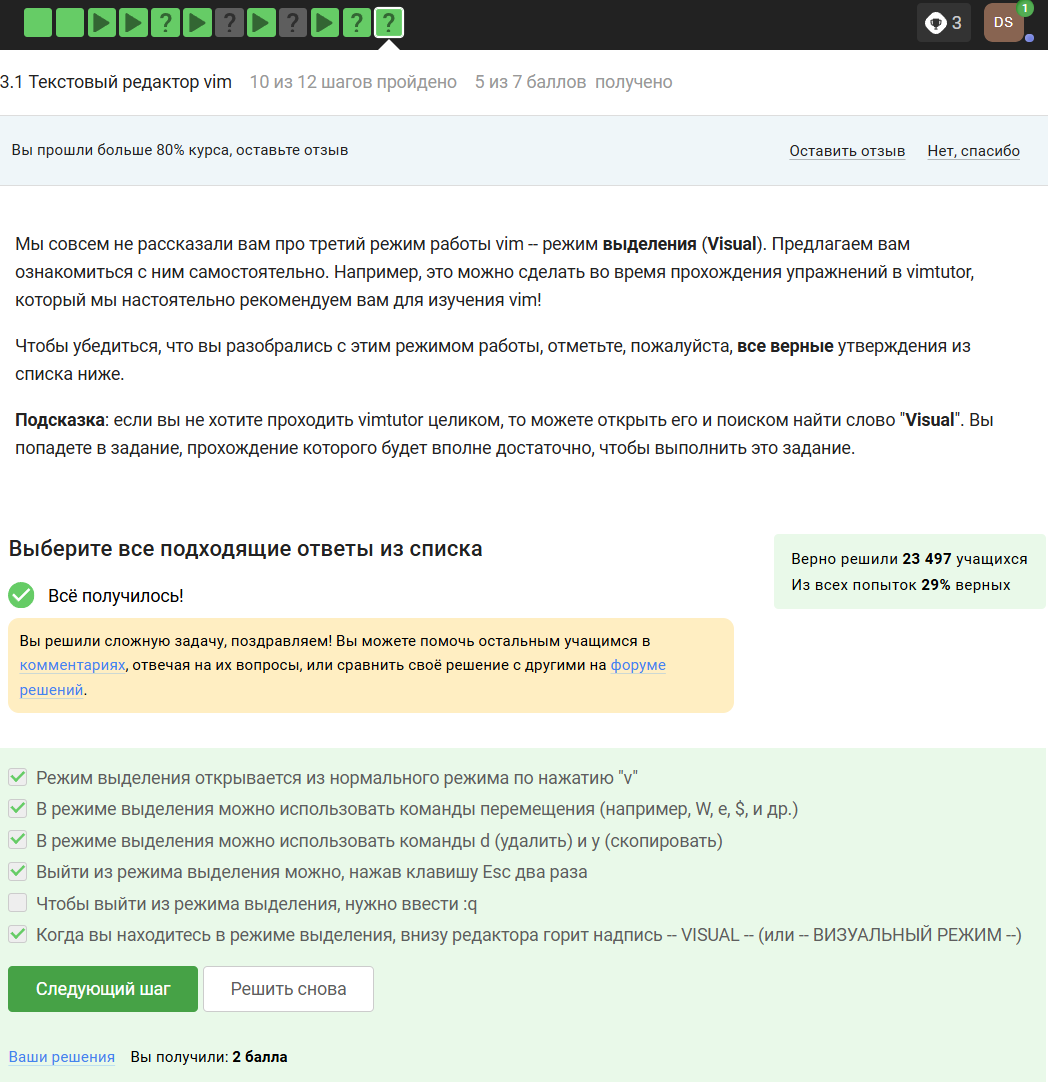


Figure 5: Задание 5

Команда $ — в конец текущей строки, W - до пробела вправо - то есть, перемещение.

Нажать Esc достаточно один раз, но да ладно.

Надпись visual - горит.

d — используется совместно с командами перемещения. Удаляет символы с текущего положения курсора до положения после ввода команды перемещения.

yy (также Y) — копирование текущей строки в буфер;

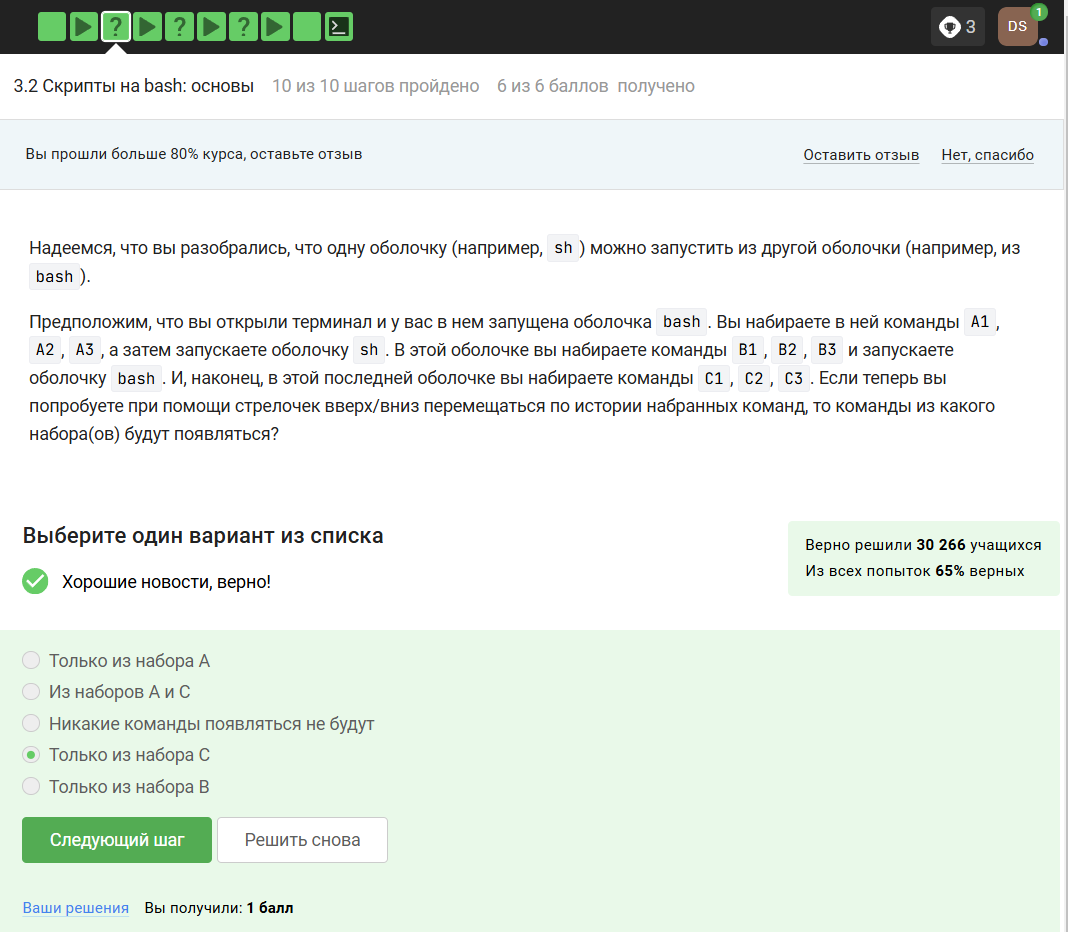


Figure 6: Задание 6

Только из набора С потому что у каждой оболочки свой буфер, который при выходе из нее буде записываться в файл истории.

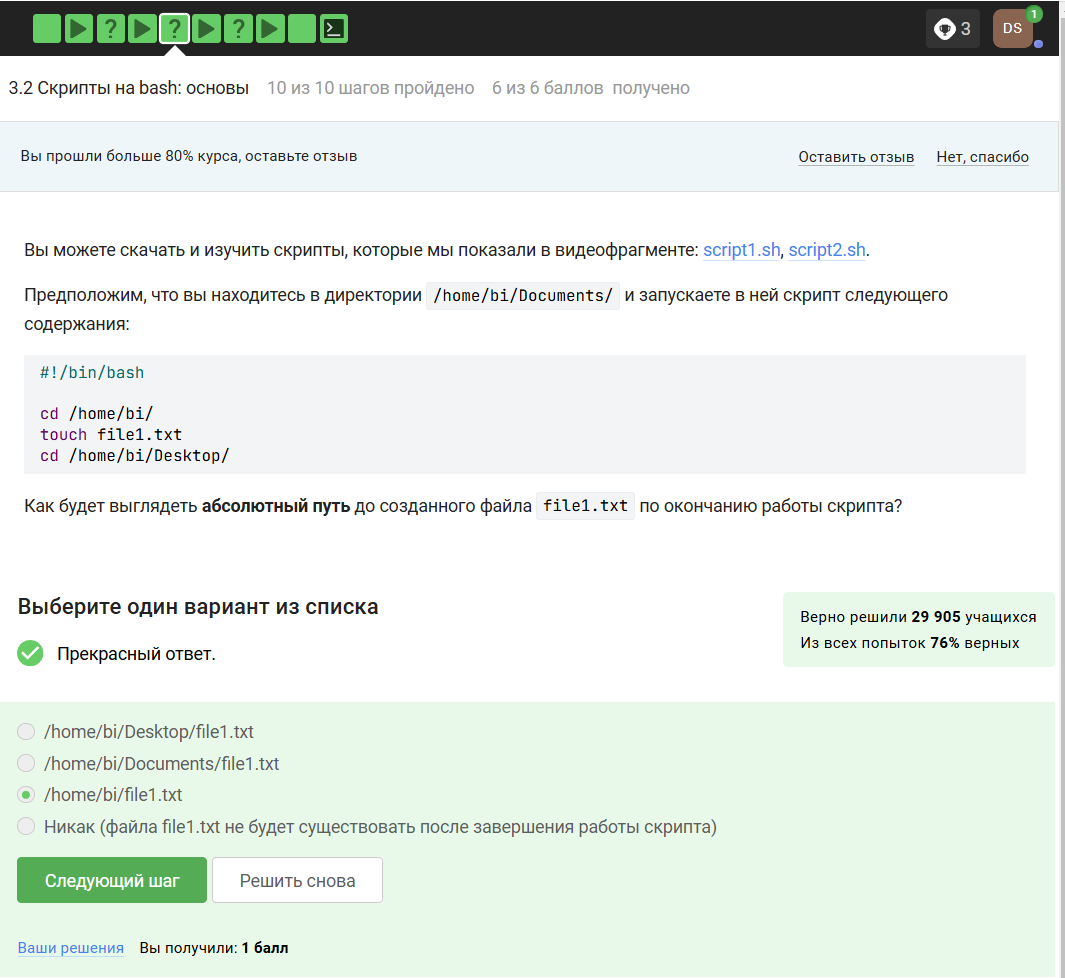


Figure 7: Задание 7

/home/bi/file1.txt - потому что именно в этой директории мы создаем новый файл, а уже после его создания мы переходим в другую папку.

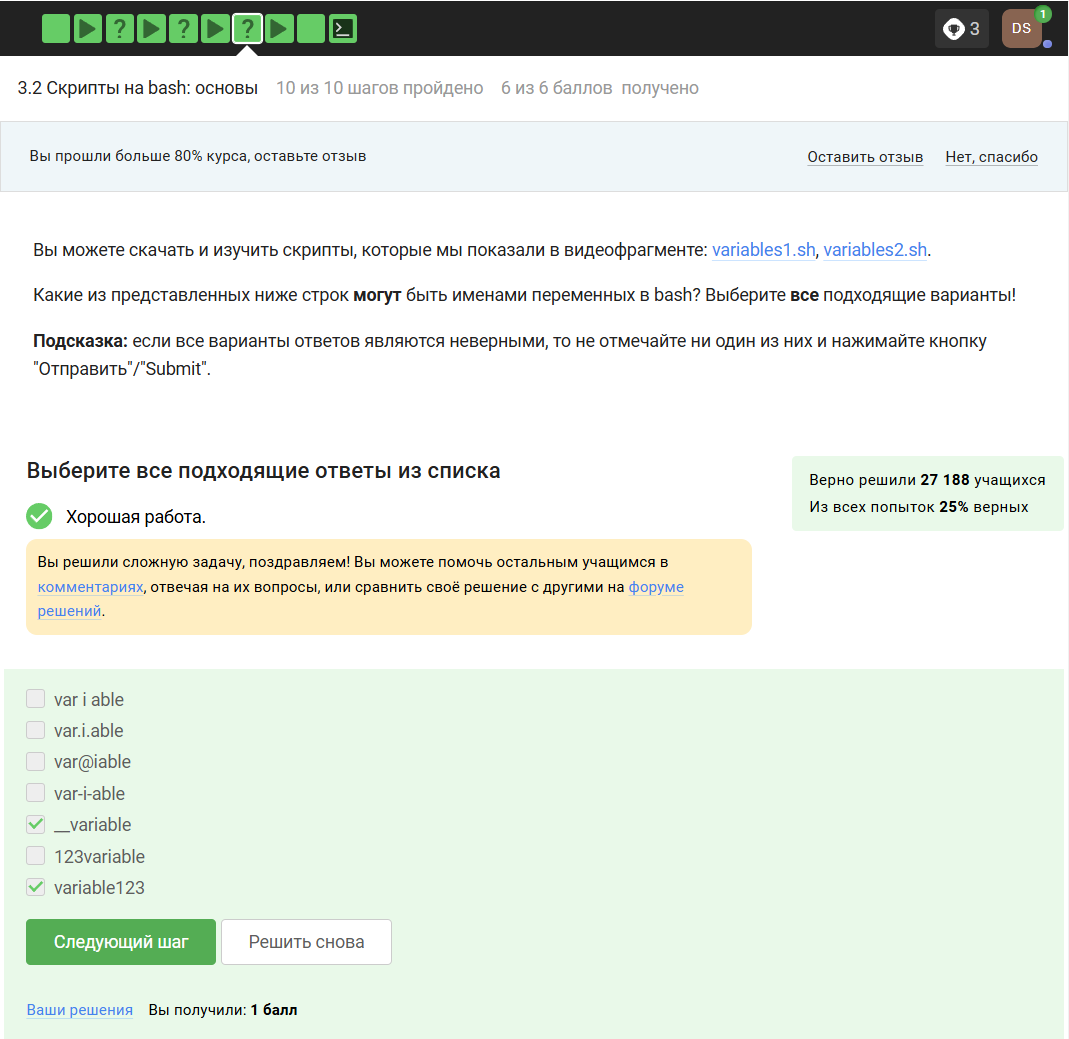


Figure 8: Задание 8

Имя не может начинаться с цифры, содержать специальные символы или пробелы.

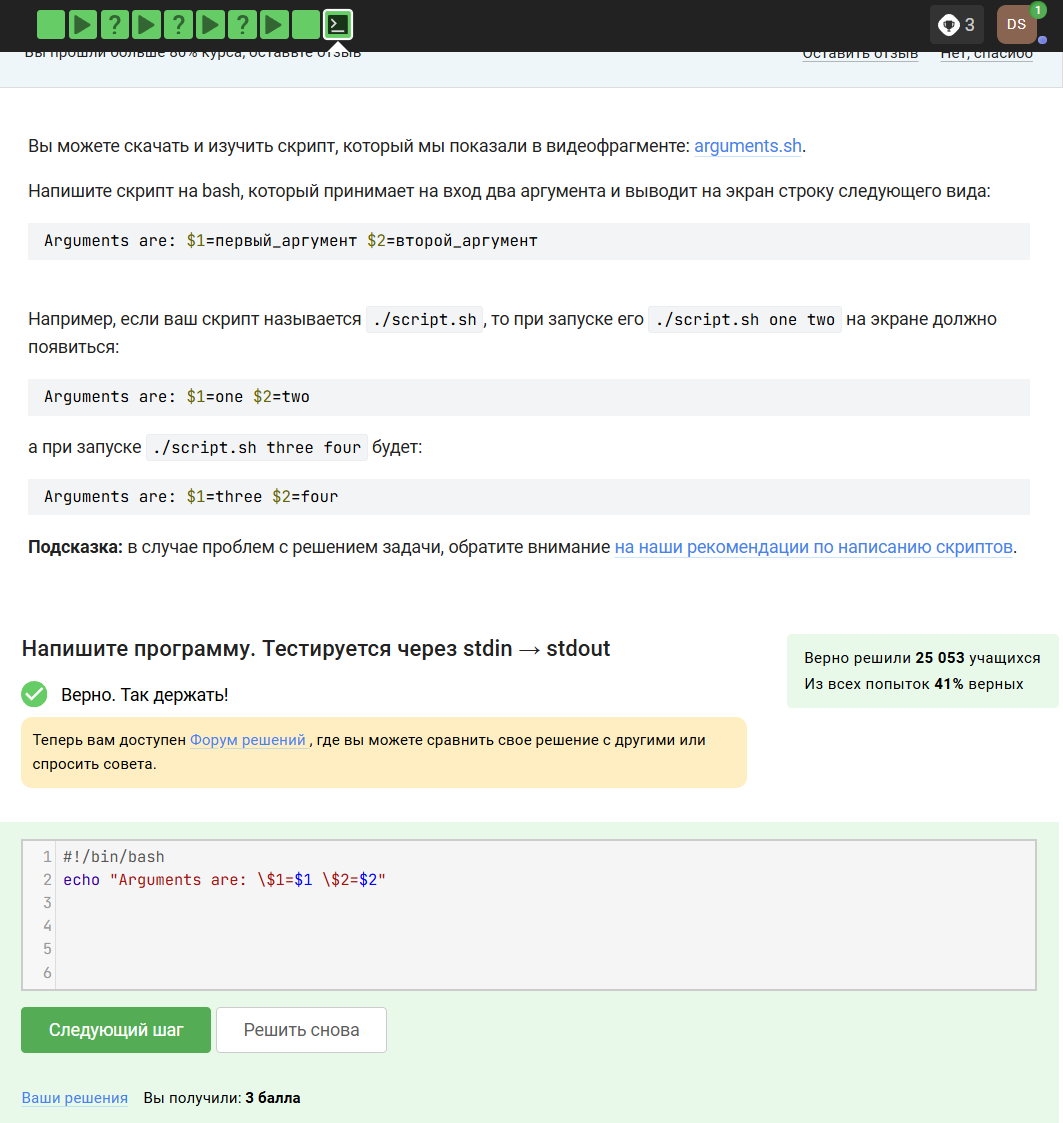


Figure 9: Задание 9

$ echo опции строка Эта команда печатает строки, которые передаются в качестве аргументов в стандартный вывод и обычно используется в сценариях оболочки для отображения сообщения или вывода результатов других команд.

#!/bin/bash

echo "Arguments are: \$1=$1 \$2=$2"

Имя скрипта - это не пустая строка.

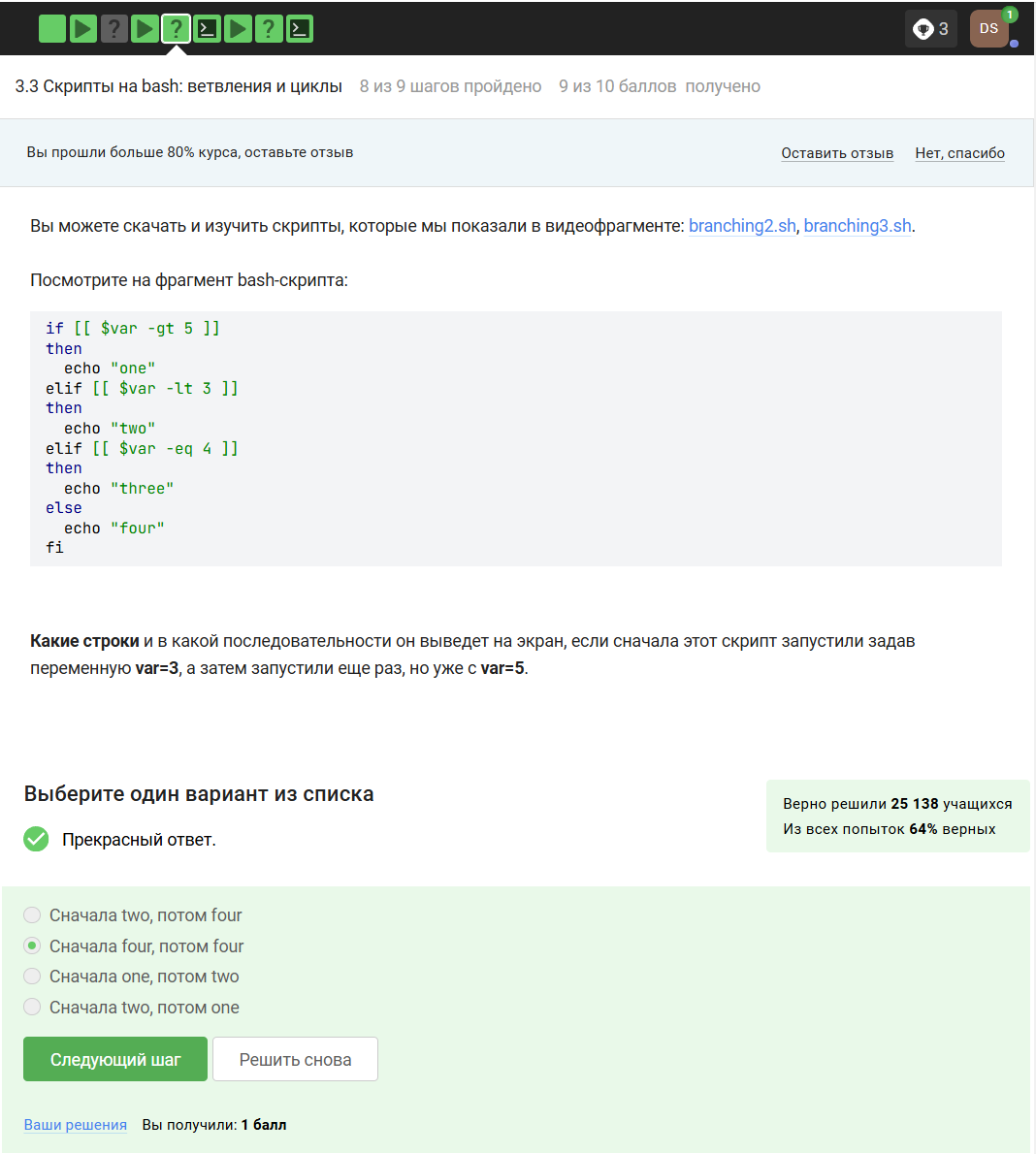


Figure 11: Задание 11

* -lt, (<) - меньше
* -gt - больше
* -eq - равно

3 не больше 5, 3 не меньше 3, 3 не равно 4.

5 не больше 5, 5 не меньше 3, 5 не равно 4.

Оба раза выведет four.

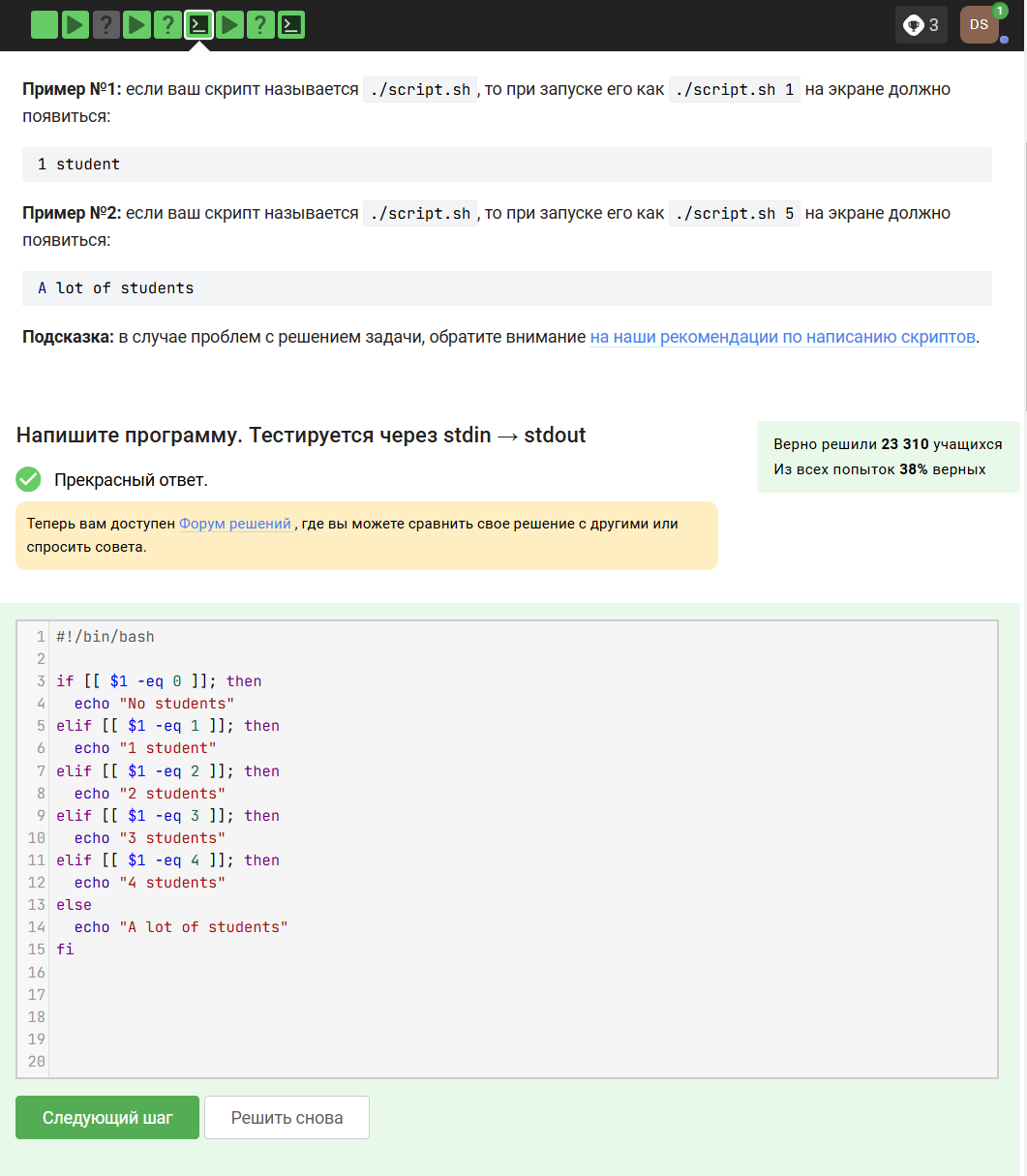


Figure 12: Задание 12

1. Задаю общую часть в каждом выводе - слово “student”: v=student
2. Выполняем команды для разных аргументов.
3. res - это результат для вывода
4. echo “$res” - вывести результат

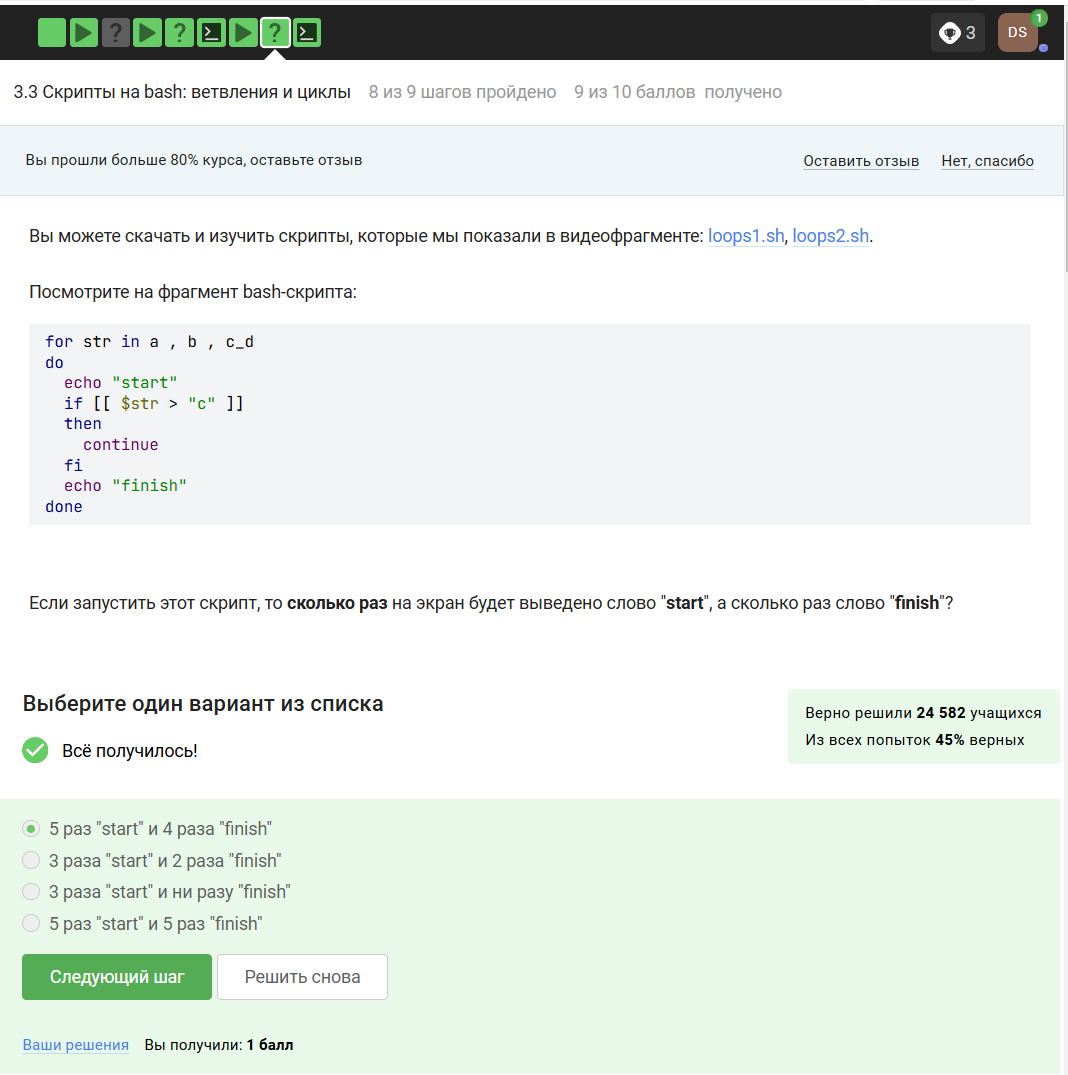


Figure 13: Задание 13

* (Start)
* a > c нет (Finish)
* (Start)
* , > c нет (Finish)
* (Start)
* b > c нет (Finish)
* (Start)
* , > c нет (Finish)
* (Start)
* c\_d > c да

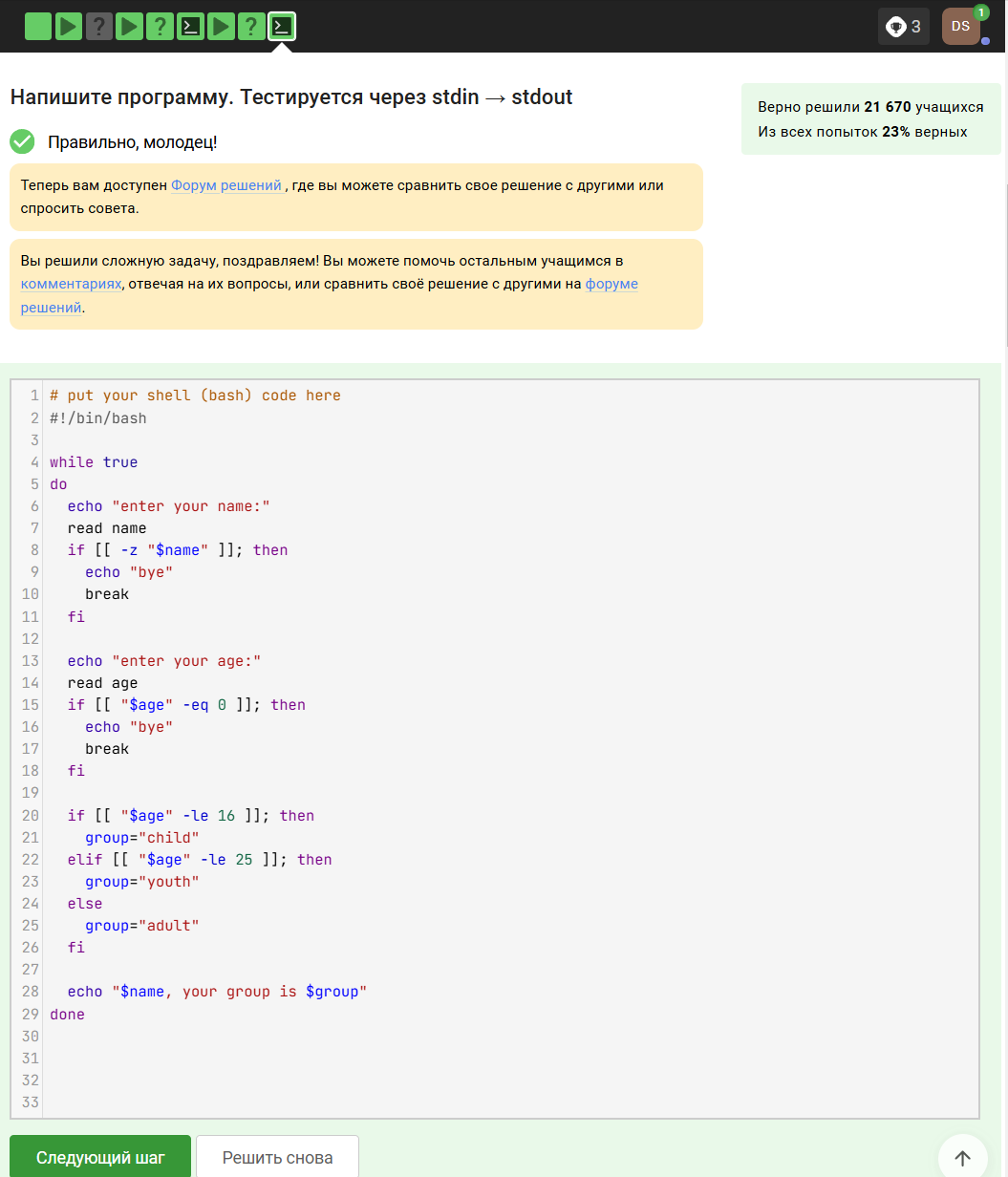


Figure 14: Задание 14

Figure 15: Задание 14

# put your shell (bash) code here

#!/bin/bash

while true

do

echo "enter your name:"

read name

if [[ -z "$name" ]]; then

echo "bye"

break

fi

echo "enter your age:"

read age

if [[ "$age" -eq 0 ]]; then

echo "bye"

break

fi

if [[ "$age" -le 16 ]]; then

group="child"

elif [[ "$age" -le 25 ]]; then

group="youth"

else

group="adult"

fi

echo "$name, your group is $group"

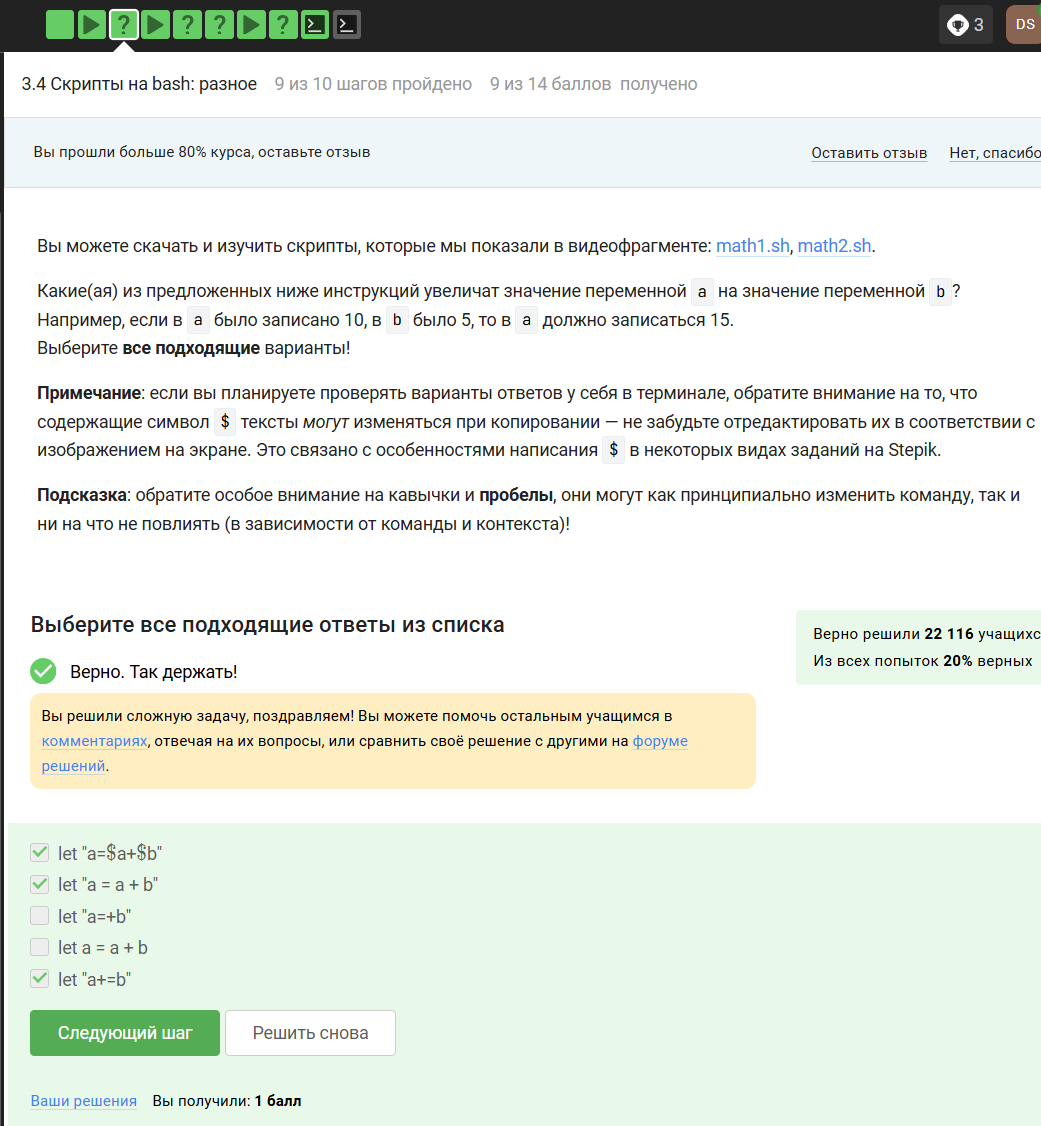
done

Figure 16: Задание 15

1. a = $a
2. a += b это то же самое, что и a = a + b, но с символами “+=” != “=+”
3. если выражение не в скобках, но с пробелами - работать не будет. (let a=a+b - сработает; let a = a + b - нет)

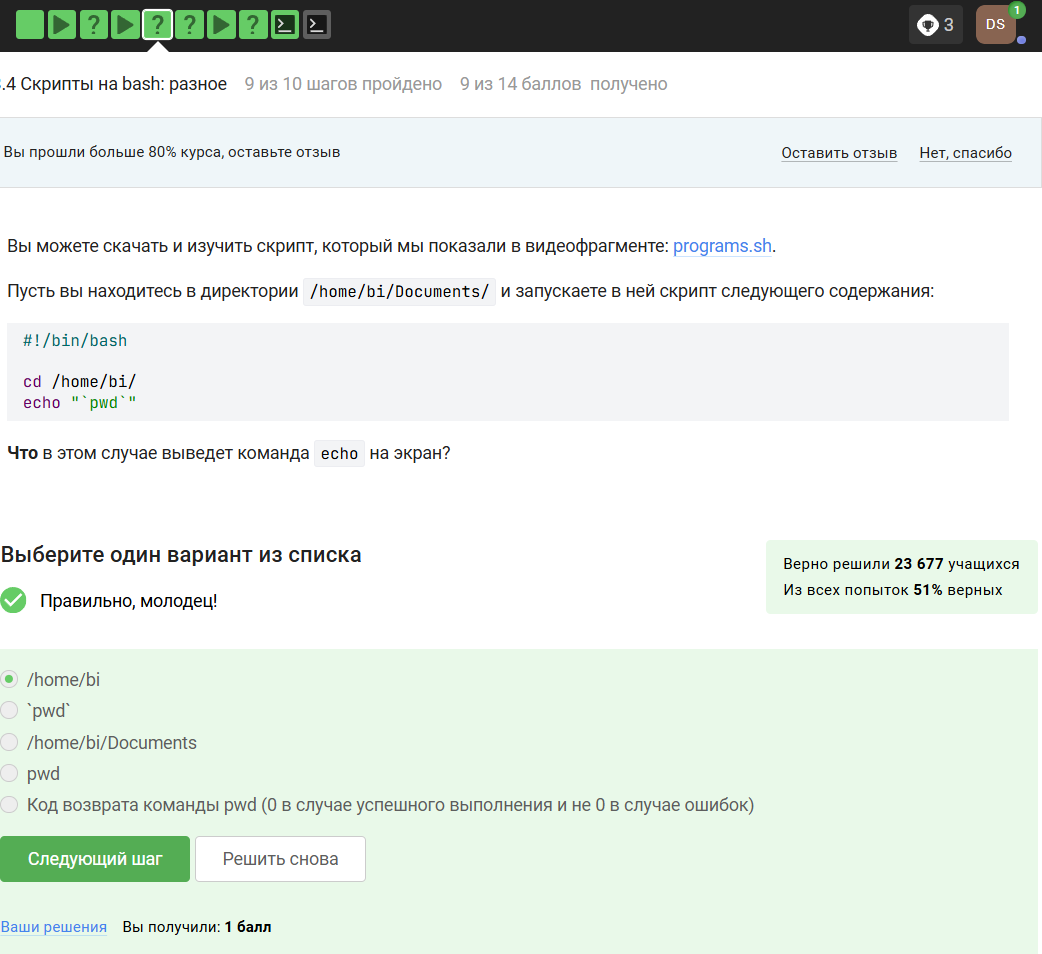


Figure 17: Задание 16

Выведет путь до директории, в которую мы перешли, так как “pwd” - это команда

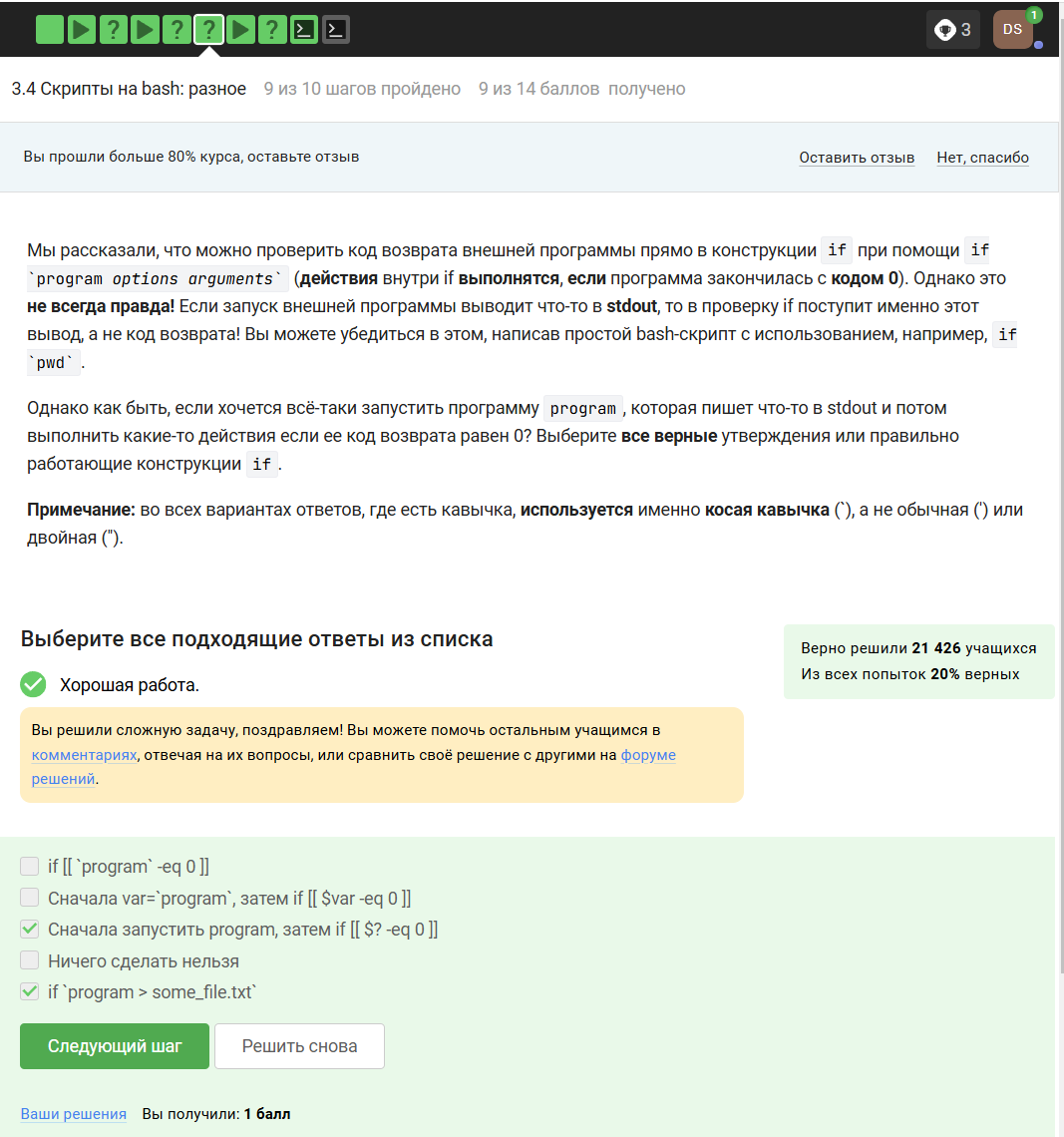


Figure 18: Задание 16\_2

programm выполняет стандартный вывод в терминал (если это принцип работы программы). И нам нужно настроить вывод в файл.

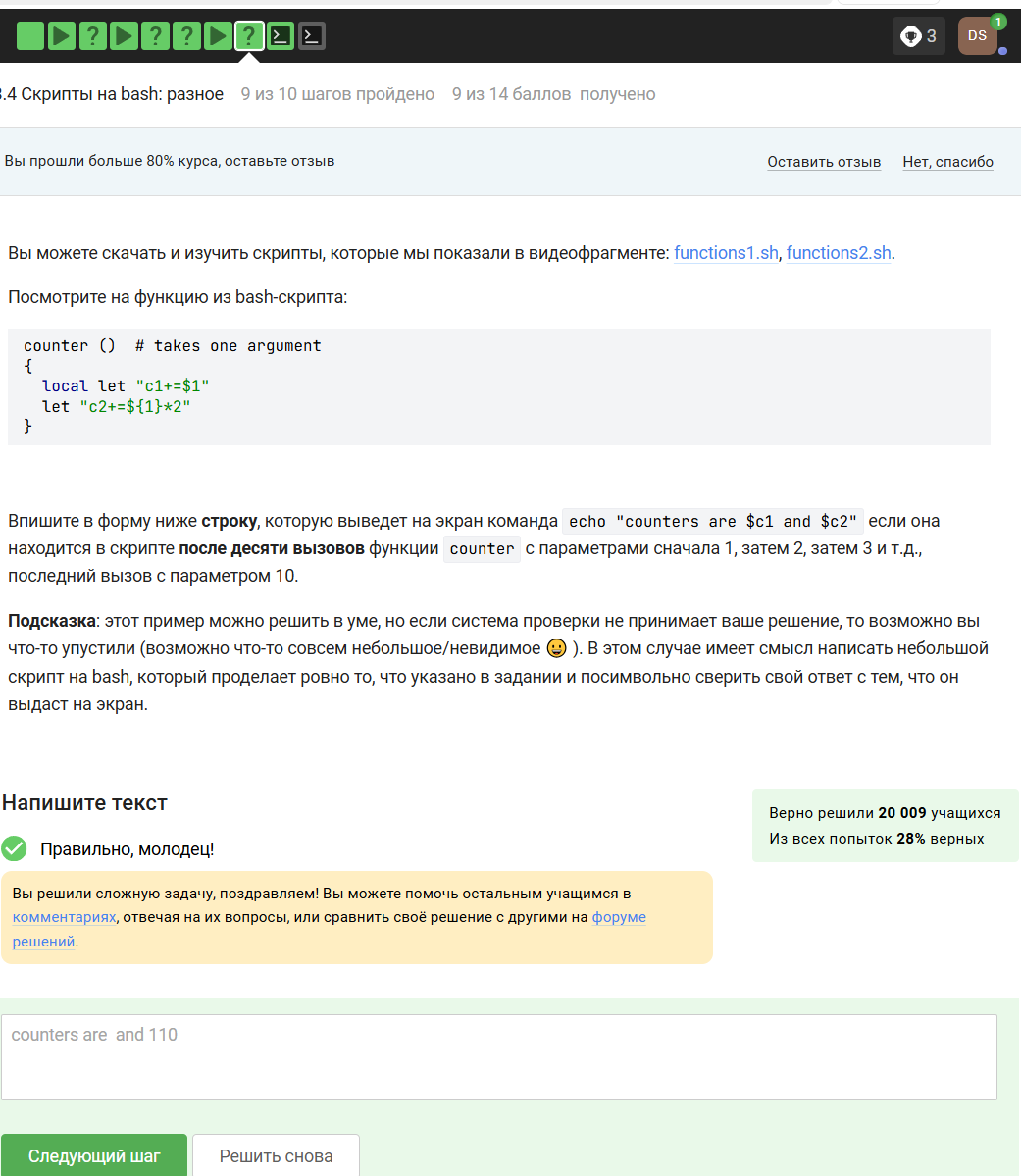


Figure 19: Задание 17

Первая переменная локальная, и это просто пустая строка, вторая переменная - это сумма арифметической прогрессии от 1 до 10, равна 55, но при умножении на 2 даст 110.

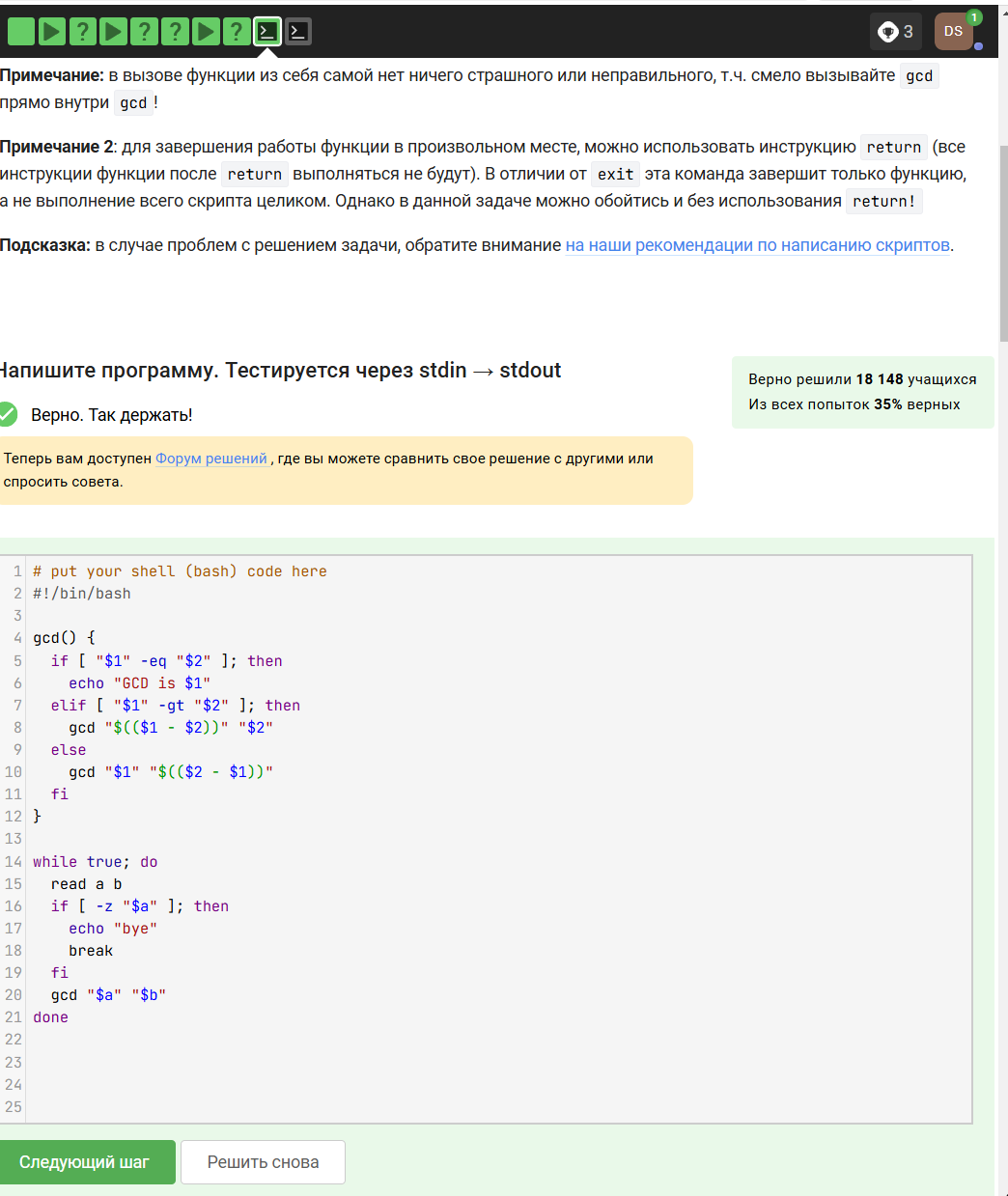


Figure 21: Задание 18

Алгоритм нахождения НОД делением

1. Большее число делим на меньшее.
2. Если делится без остатка, то меньшее число и есть НОД (следует выйти из цикла).
3. Если есть остаток, то большее число заменяем на остаток от деления.
4. Переходим к пункту 1.

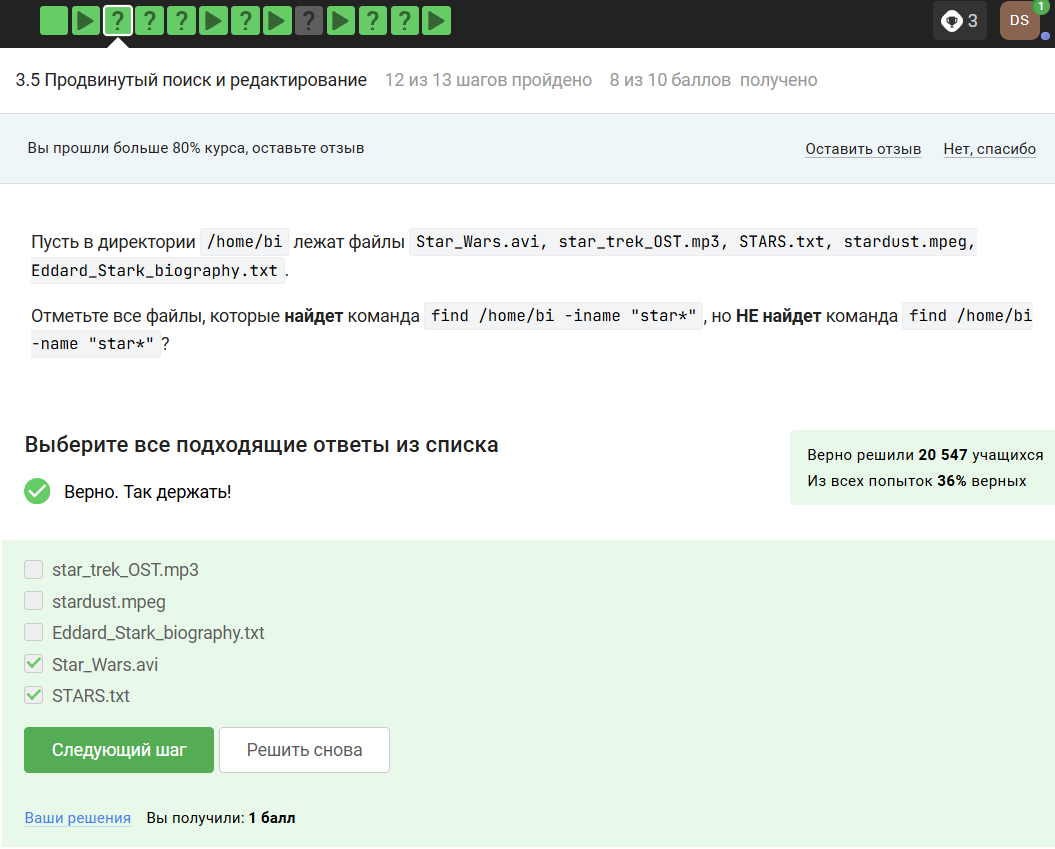


Figure 24: Задание 20

-iname ищет без учета регистра, а -name в точности как в запросе. Звездочка стоит после слова - это значит после слова может быть сколько угодно символов.

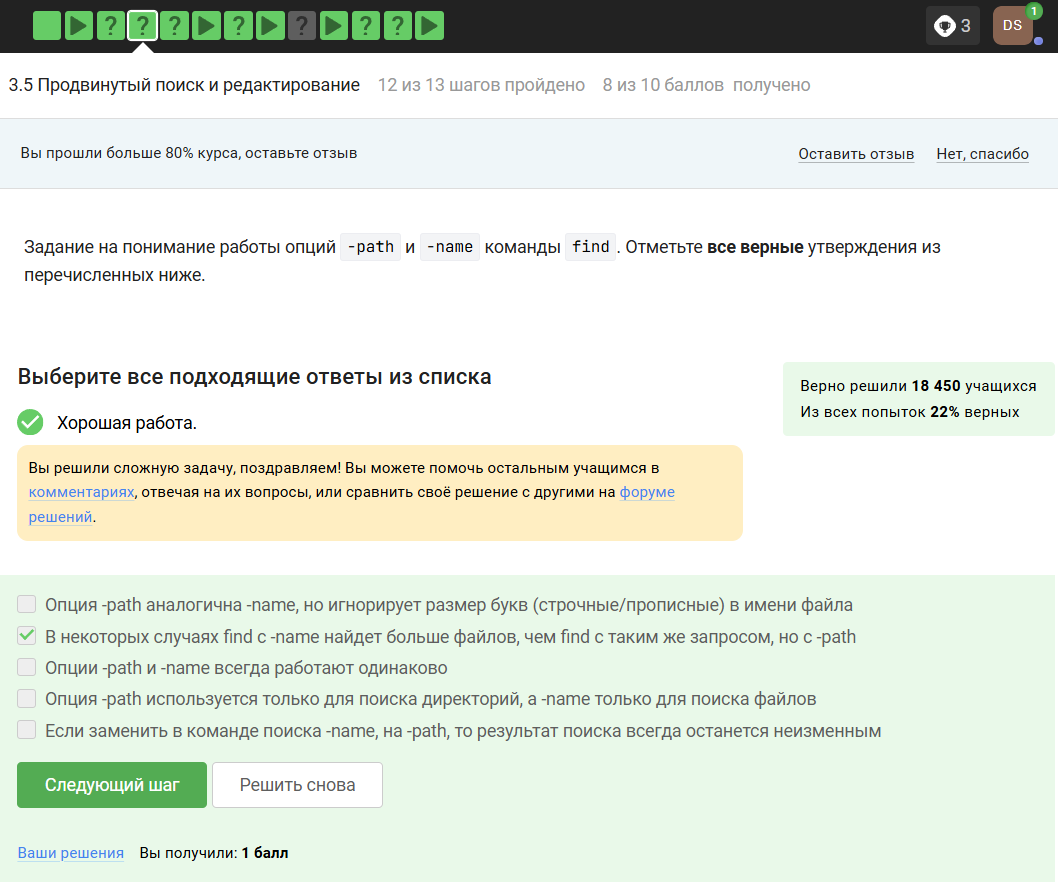


Figure 25: Задание 21

find [path] [expression]

где: path - это путь к директории, в которой нужно выполнить поиск файлов (по умолчанию, поиск производится в текущей директории и всех ее поддиректориях);

expression - это выражение, которое определяет критерии поиска файлов.

-name: поиск файлов по имени. Например: find /home/user -name myfile.txt

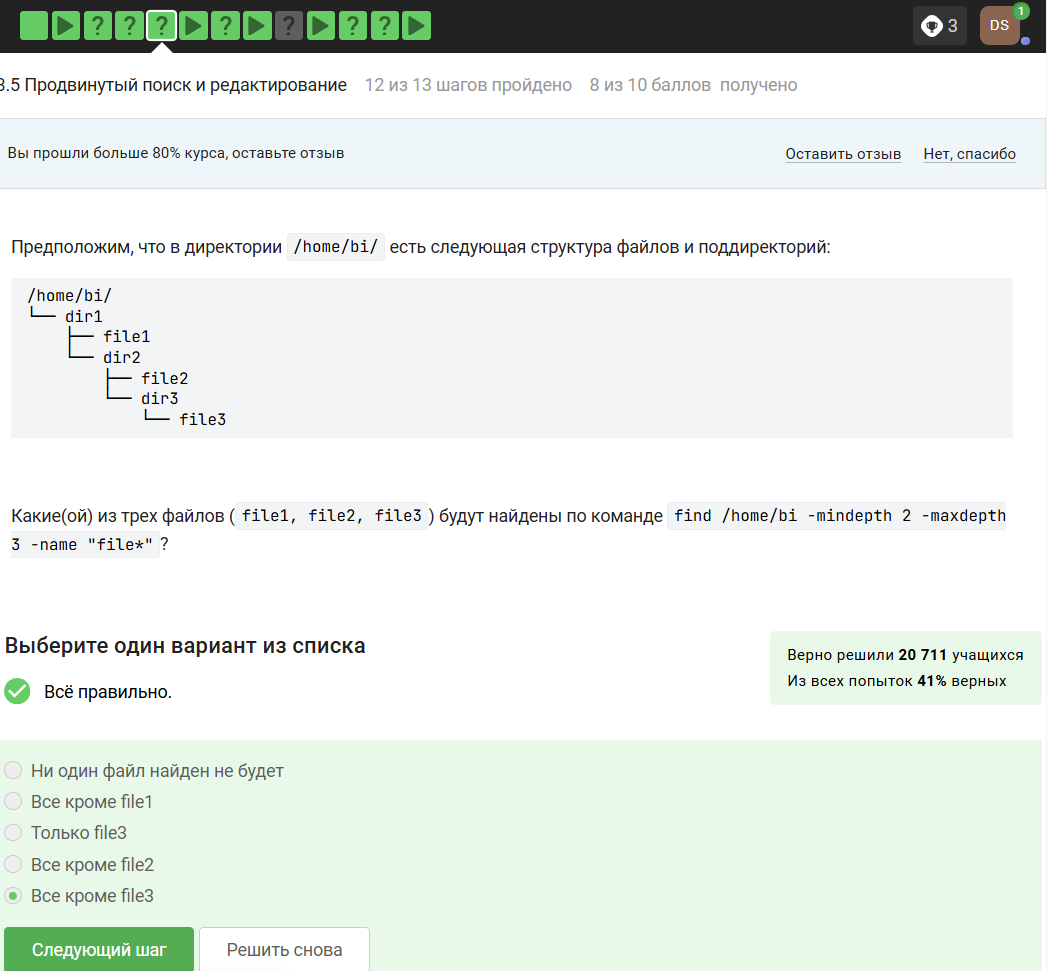


Figure 26: Задание 22

Текущий каталог - это depth=1, а остальное считается просто:

/home/bi -> depth=1

/home/bi/dir1 -> depth=2

/home/bi/dir1/dir2 -> depth=3

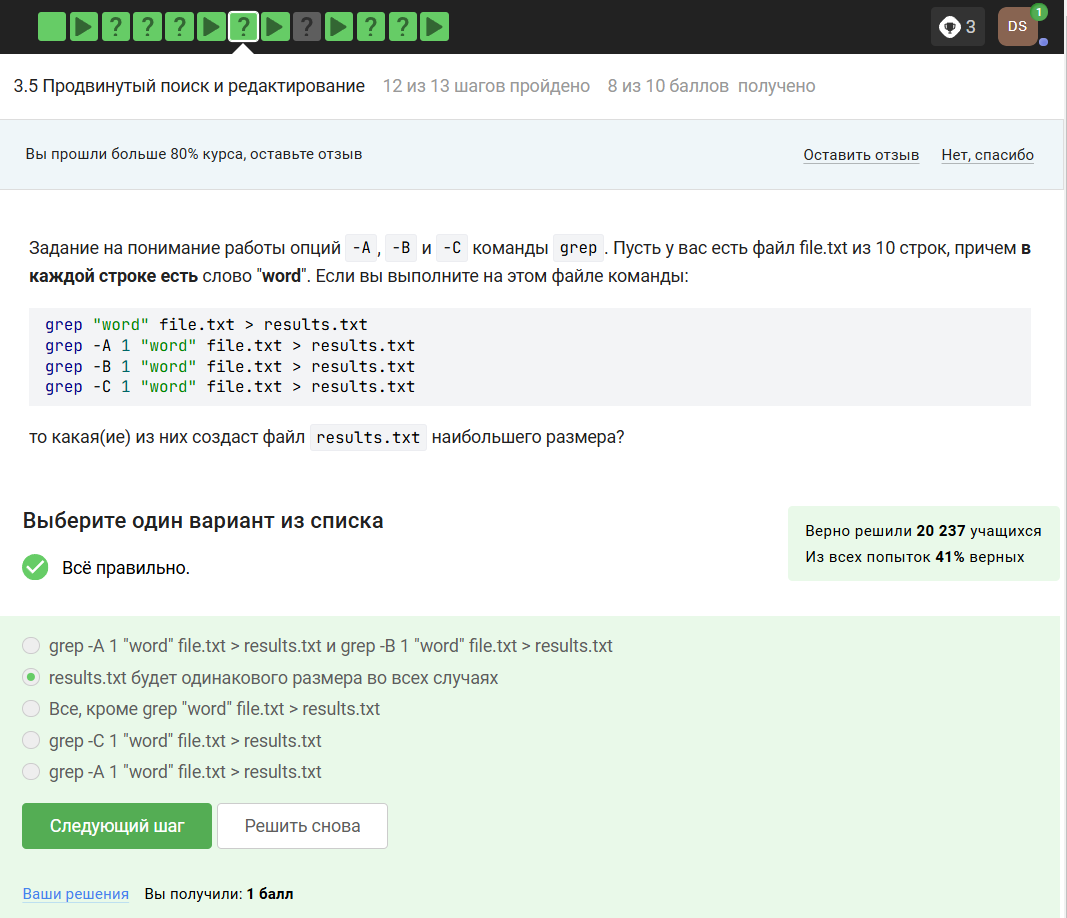


Figure 27: Задание 23

Из описания man: Print NUM lines of trailing context after/before matching lines “matching lines” - множественное число, строки в которых нашлось совпадение

Т.е. если идут 2…10…100 строк подряд, в которых обнаружилось совпадение, контекст будет выведен до и после этой ГРУППЫ строк, а не до и после каждой строки в этой группе

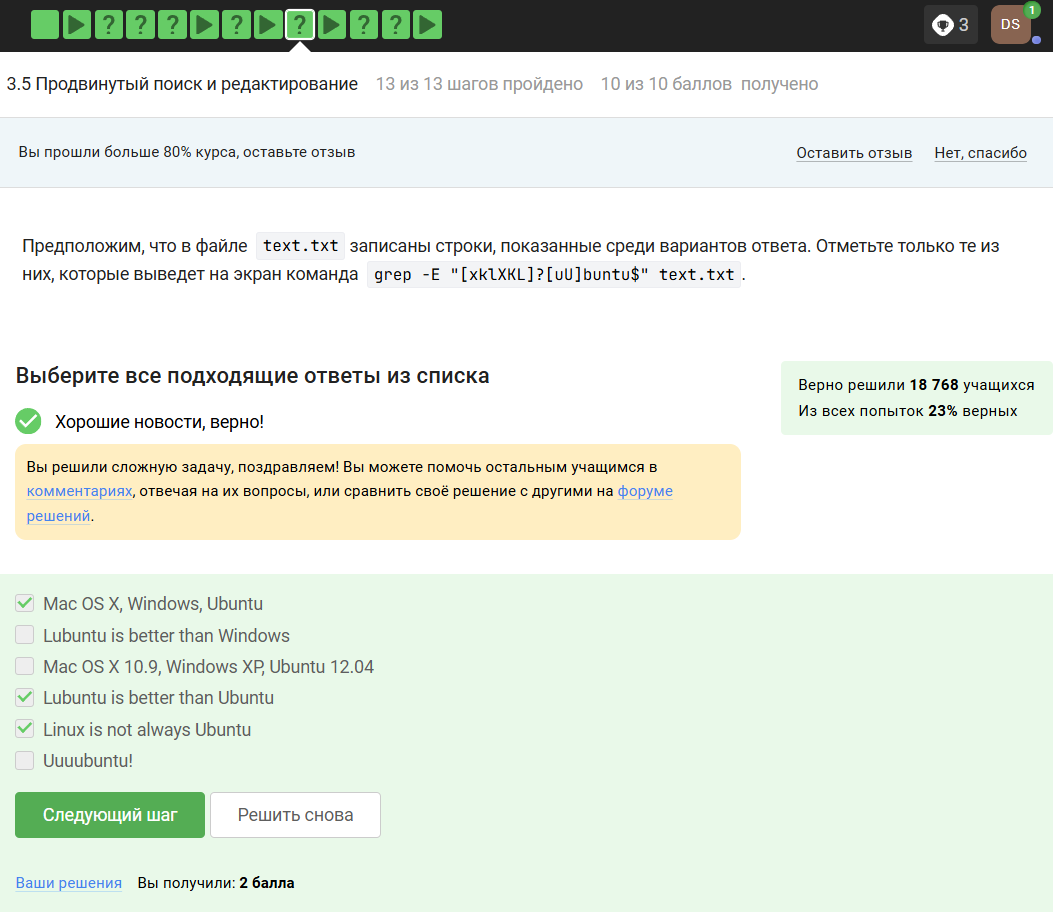


Figure 29: Задание 24

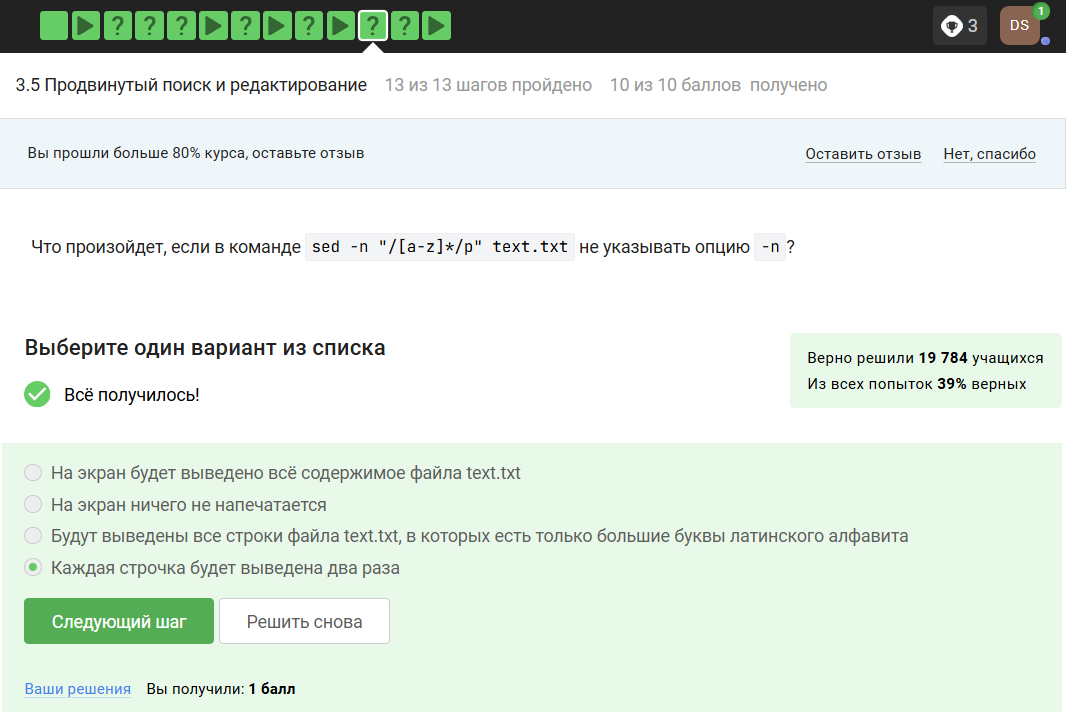


Figure 30: Задание 25

The -n option disables the automatic printing, which means the lines you don’t specifically tell it to print do not get printed, and lines you do explicitly tell it to print (e.g. with p) get printed only once.

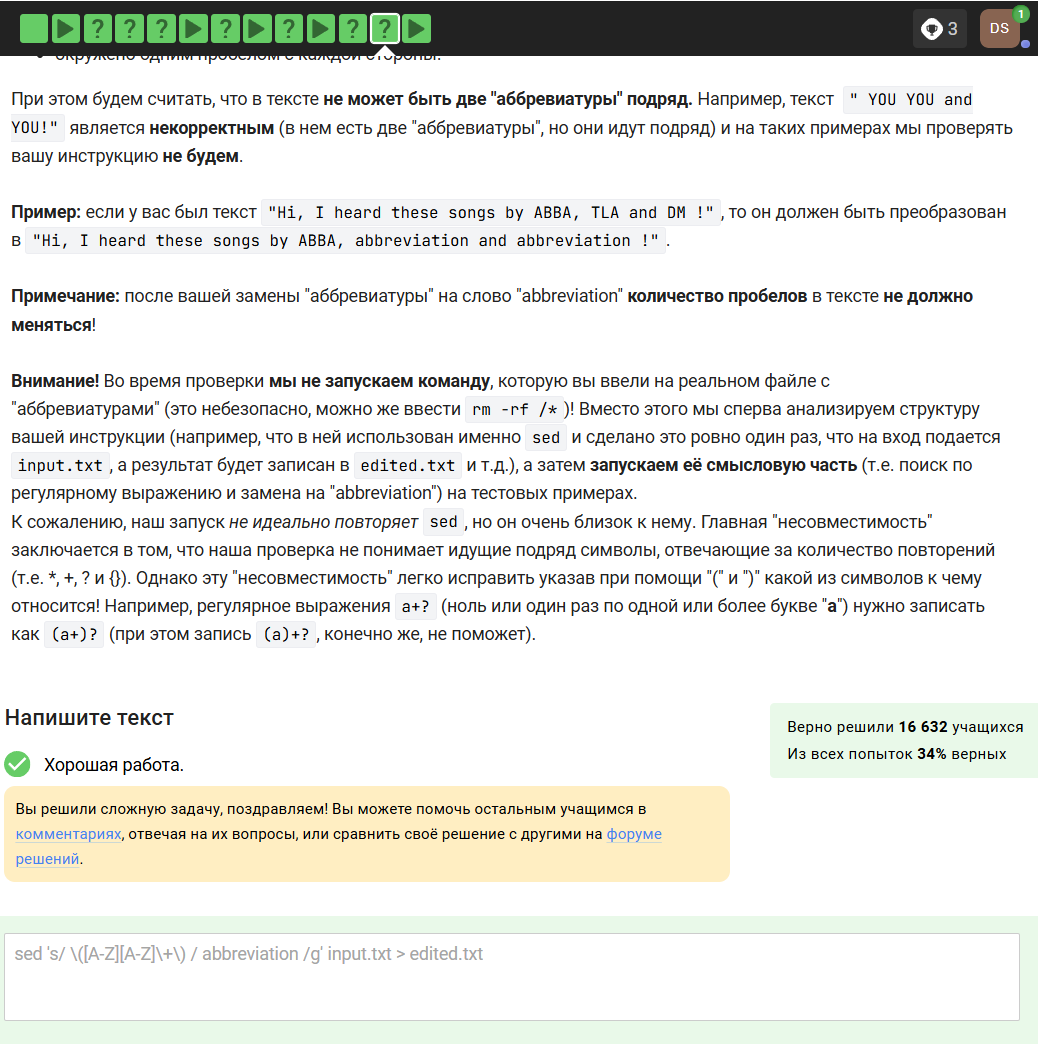


Figure 31: Задание 26

аббревиатура ABBA отличается от двух других аббревиатур тем, что справа он неё стоит запятая без пробела: “ABBA,”.

При этом по условию аббревиатура должна выглядеть как [ XX ] или [ XXX ] (и ещё больше X). Следовательно, для этой проверки надо добавить пробел квадратными скобками [ ] слева и, соответственно, с права.

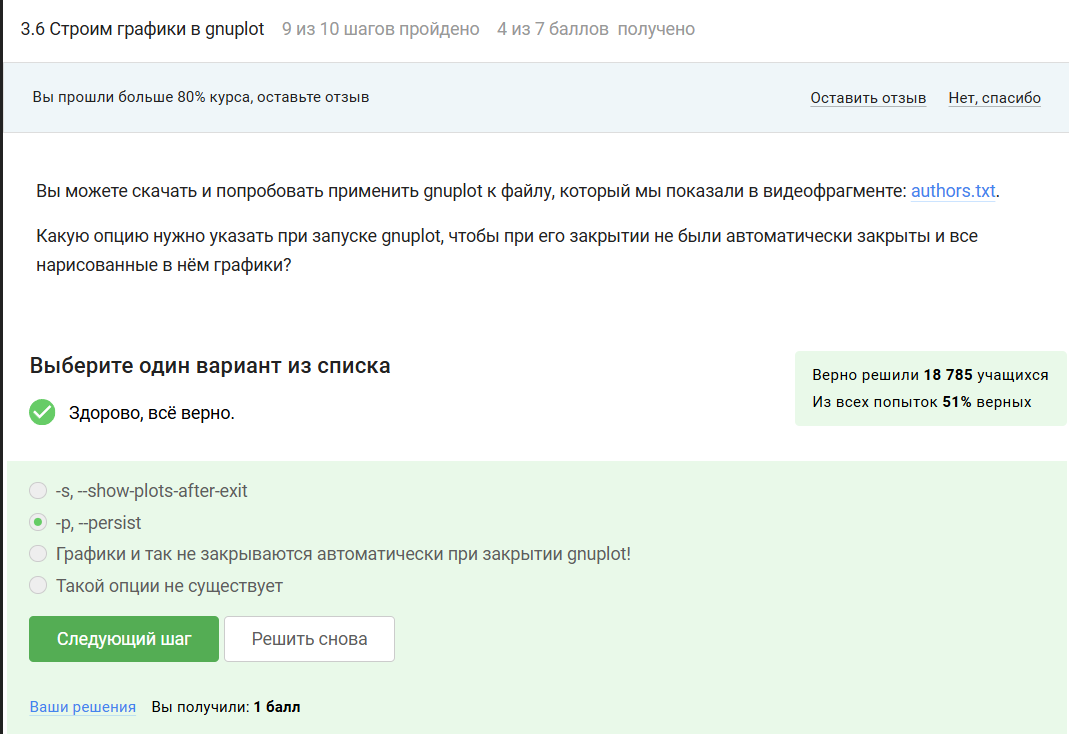


Figure 32: Задание 27

-persist lets plot windows survive after main gnuplot program exits.

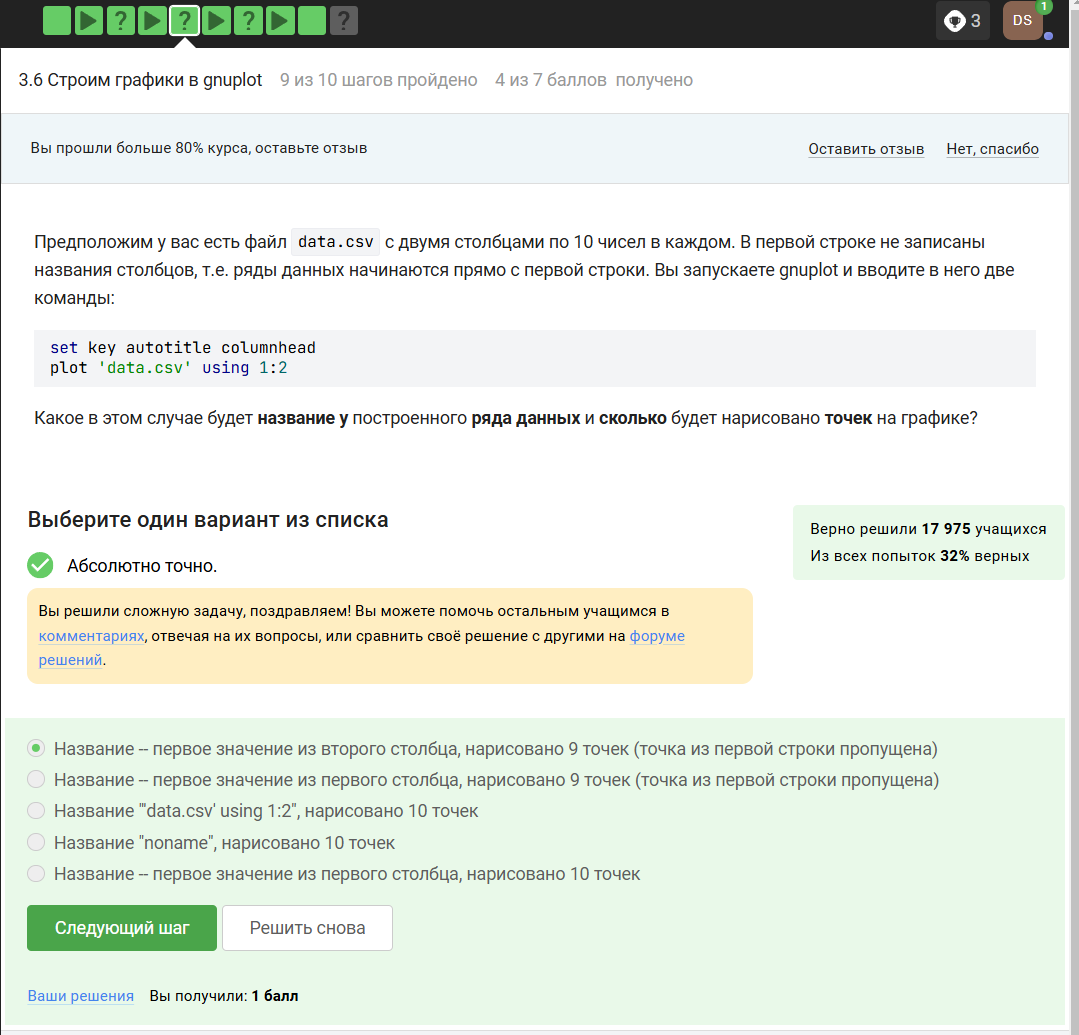


Figure 33: Задание 28

plot 'data.csv' using 1:2 даст ошибку:

warning: Skipping data file with no valid points ^ x range is invalid

Скорее всего причиной такого поведения является тот факт, что формат CSV содержит строки, где столбцы разделены запятой? Содержимое файла:

1,21  
2,22  
3,23  
4,24  
5,25  
6,26  
7,27  
8,28  
9,29  
10,30

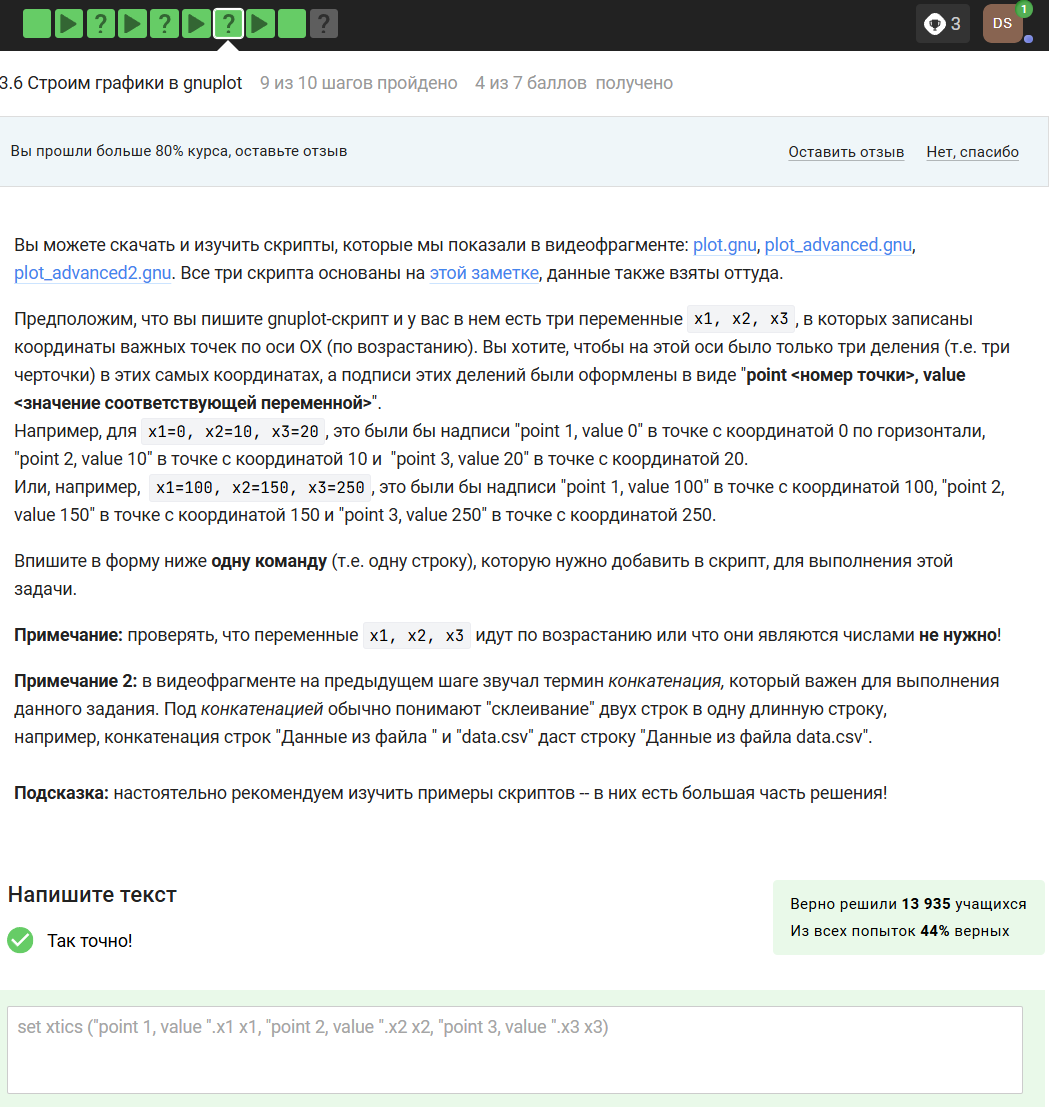


Figure 34: Задание 29

Cначала идет команда установки подписей, а потом в скобках:

подпись - пробел - переменная с координатой - запятая

Повторяется это количество раз соответствующее числу переменных, и без запятой (в случае с последней переменной)

А подпись в свою очередь получается конкатенацией текста из задания и переменной с координатой.

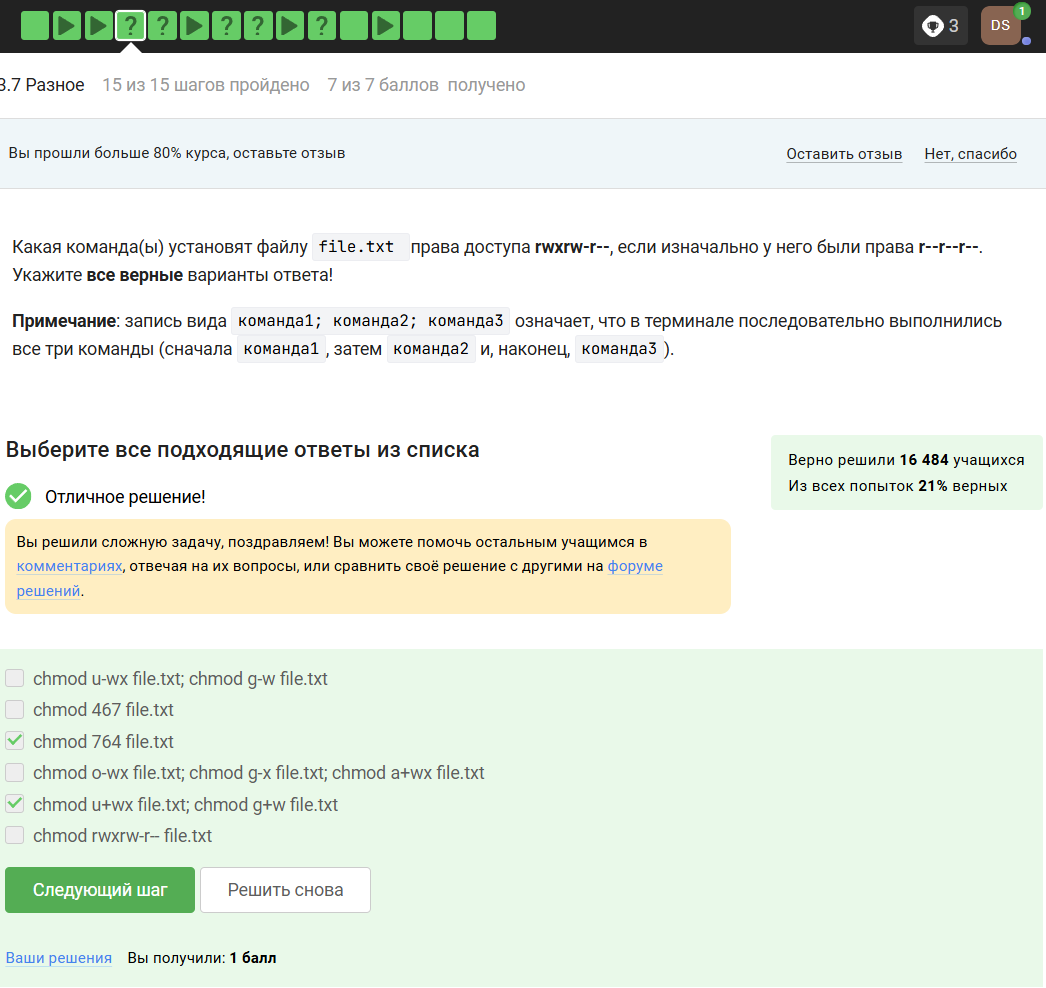


Figure 36: Задание 31

* r - чтение;
* w - запись;
* x - выполнение;
* s - выполнение от имени суперпользователя (дополнительный);
* u - владелец файла;
* g - группа файла;
* o - все остальные пользователи;
* 0 - никаких прав;
* 1 - только выполнение;
* 2 - только запись;
* 3 - выполнение и запись;
* 4 - только чтение;
* 5 - чтение и выполнение;
* 6 - чтение и запись;
* 7 - чтение запись и выполнение.

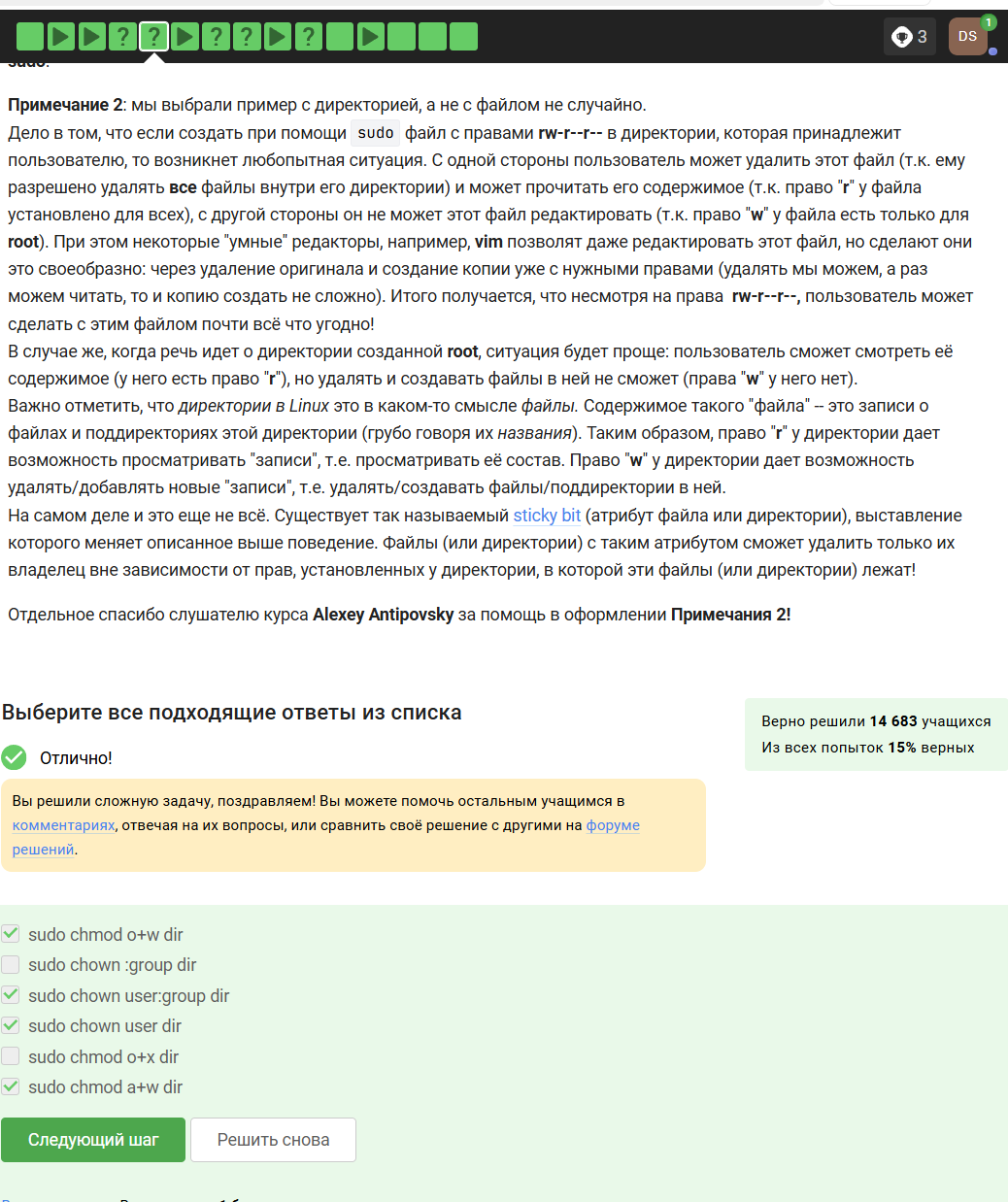


Figure 37: Задание 32

Решений два типа:

* Сменить права гостей, добавив W
* Сделать владельцем нужную группу или пользователя, в зависимости от того, у кого из них уже есть права на W
* Помнить, что root - владелец и остальные для него - others.

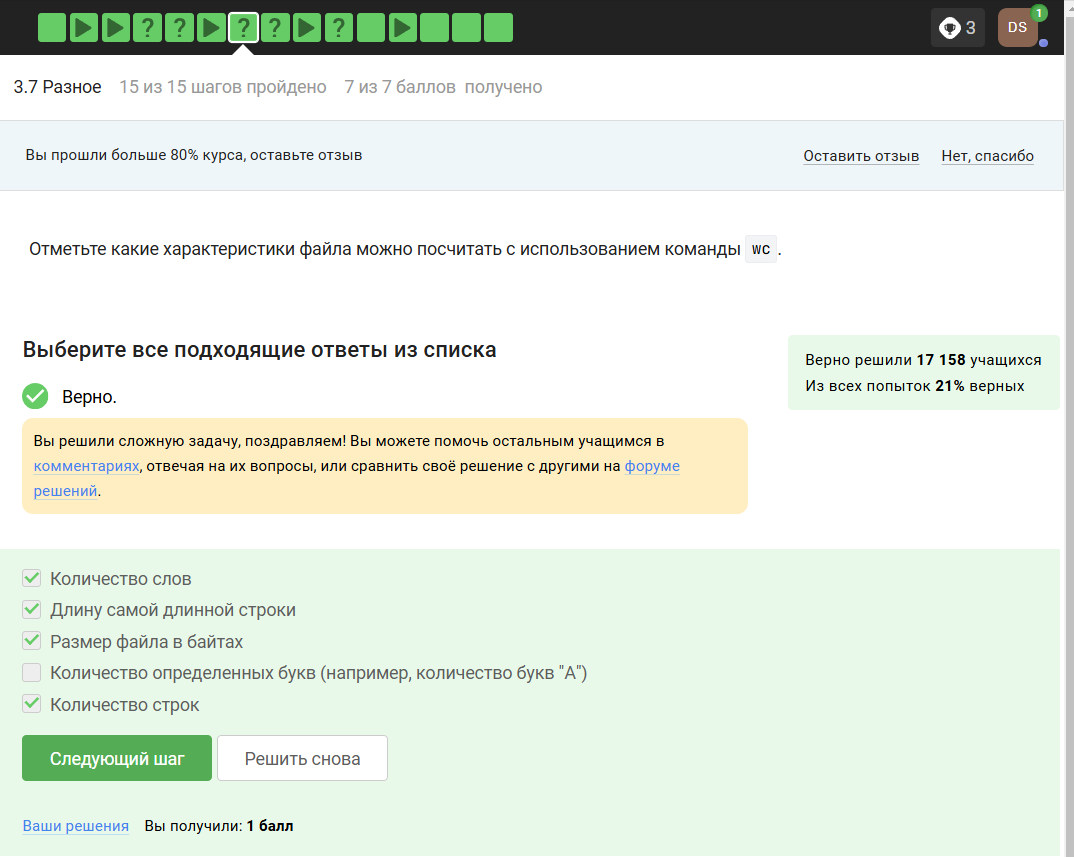


Figure 38: Задание 33

* wc -l вывести количество строк
* wc -c вывести количество байт
* wc -m вывести количество символов
* wc -L вывести длину самой длинной строки
* wc -w вывести количество слов

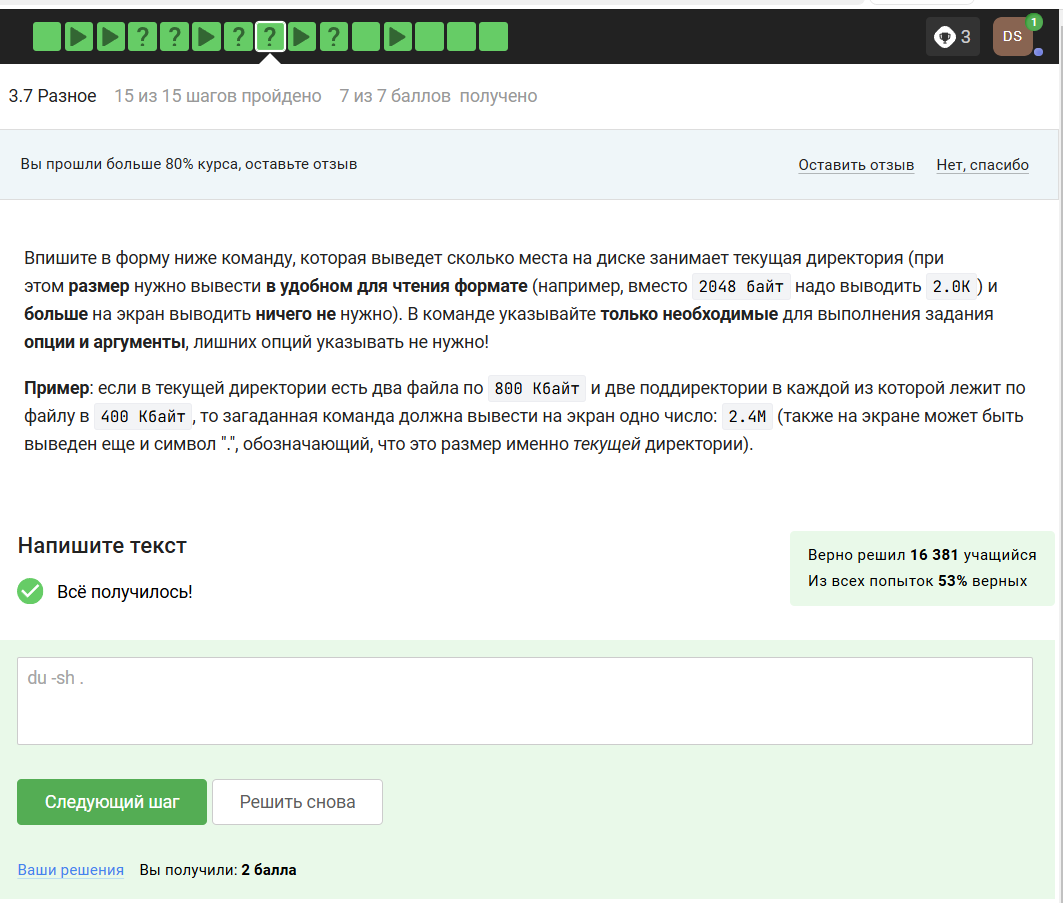


Figure 39: Задание 34

-h, –human-readable print sizes in human readable format (e.g., 1K 234M 2G)

-s, –summarize display only a total for each argument

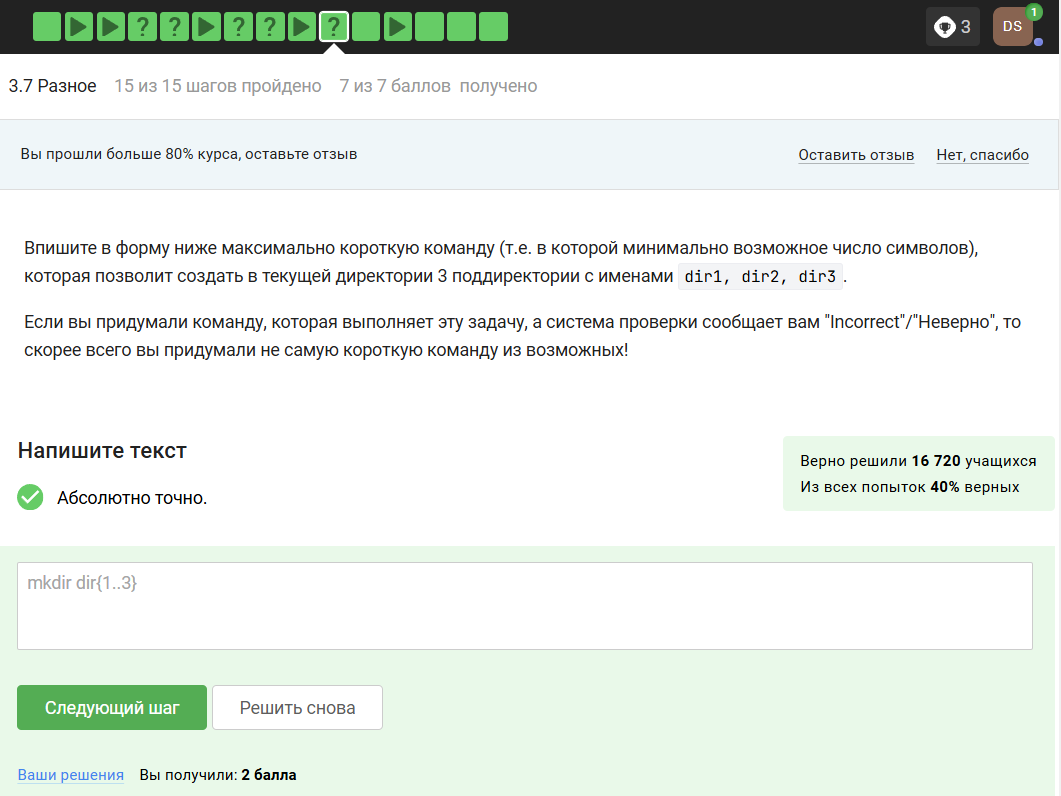
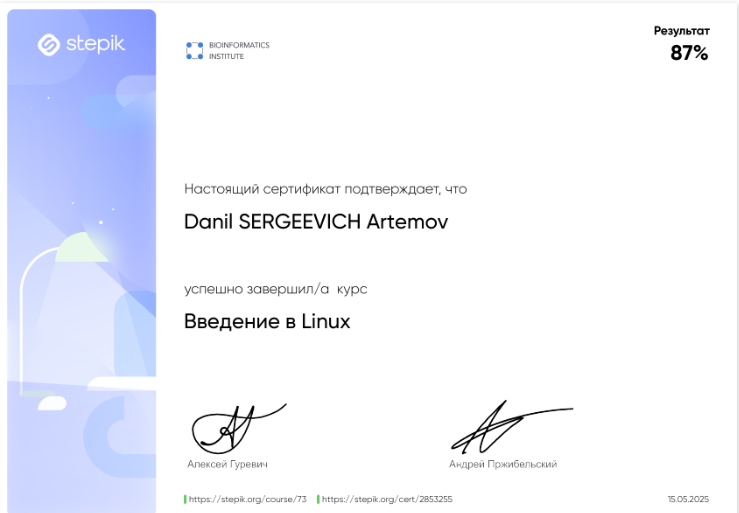


Figure 40: Задание 35

Команда создаст три директории от dir1 до dir3.

# 5 Сертификат

Figure 41: Сертификат



# 6 Выводы

Я просмотрела курс и освежила в памяти навыки работы с более сложными командами в Линукс.

# Список литературы

1. Введение в Linux