Tourists Pairing

Tourism Development Corporation of Bali provides transportation services to groups of tourists traveling to the same destination for sightseeing. They have a fleet of different cars available which have different seating capacities respectively. For a given group of tourists, they want to calculate the number of vehicles required to efficiently accommodate all tourists. Since it's a free-of-cost service and all the tourists are traveling to the same destination, the company wants the vehicles to be filled up to their maximum capacity and they want to use as few vehicles as they can to minimize the operational cost.

There are two arrays given as input;

- T[ ] which represents the tourist array that contains the number of tourists in each group
- S[ ] which represents the seating capacities array that contains the number of seats available in each vehicle

Write a function that, given two arrays T[ ] of tourists and S[ ] of seating capacities, consisting of integer elements, returns the minimum number of vehicles needed to accommodate all the passengers.

Given:

- Each element of arrays T and S is an integer within the range [1....100]
- T[K] <= S[K] for each K within the range [0..N-1]

Input

The input will be read from a file. The filename/path is passed to your program as the first command-line argument. The file will have two integer arrays, T[ ] and S[ ]. The first line of the file will contain a list of tourists and the second line will contain a list of seating capacities in each vehicle.

Output

The output will be an integer. (Minimum number of vehicles needed)

Examples:

Example 1. Given T = [4, 4, 2, 4] and S = [5, 5, 2, 5], the function should return 3.

Explanation:

5 tourists from group number 0 and 1 can travel in car number 0 with 5 seating capacity.
5 tourists from group number 1 and 2 can travel in car number 1 with 5 seating capacity.
The remaining 4 tourists can travel in car number 3 with 5 seating capacity.
Example 2. Given P = [2, 3, 4, 2] and S = [2, 5, 7, 2], the function should return 2.

Explanation:

5 tourists from group number 0 and 1 can travel in car number 1 with 5 seating capacity.
6 tourists from group number 2 and 3 can travel in car number 2 with 7 seating capacity.

ShareDrive

ShareDrive, a new ride sharing service, has recently started its operations in Pakistan. To attract more drivers to their app, they are offering a joining package for drivers. Some of the key features in offer include:

- First month is completely free for drivers
- 10% commission on earnings for the second month i.e if driver is earning 1000 Rs in second month, he will pay 100 rupees to Sharedrive.
- and 20% commission on next months i.e if driver is earning 1000 Rs in next month, he will pay 200 rupees to Sharedrive.

In order to pay back the commission amount to Sharedrive, drivers can pay back any amount via Easypaisa or Jazzcash. All the payments are available in Payments-API below. If the amount paid is more than the required commission amount, It will become part of their balance and can be adjusted later.

ShareDrive requires your help in calculating the total amount that a driver owes or needs to be paid by the company in 2022. You are provided with 2 API endpoints for all the data you need:

Rides API: https://www.jsonkeeper.com/b/DM5F

This API will return all the trips in JSON format and each trip contains driver id , fare and trip date that can help you calculate the required commission amount.

Date format is MM/DD/YYYY

Format: JSON. List of trip objects, one object looks like this

```
{
  "driver_id": 10006,
  "driver_name": "Herman Lowe",
  "trip_rating": 5,
  "trip_details": {
    "distance_km": 4,
    "time_taken": 17,
    "fare": 68,
    "customer_id": "0271707666"
  },
  "trip_date": "05/01/2022",
  "vehicle": {
    "model": "Galant",
    "number": "1G4CU541034935075"
  }
}
```

Payments API: https://www.jsonkeeper.com/b/9QRZ

This API will return all the payments made by the drivers to Sharedrive. Driver id and amount value in each record can help you calculate the total amount that has been paid back to Sharedrive.

Date format is MM/DD/YYYY

Format: JSON. List of payment objects, one object looks like this

```
{
  "driver_id": 10007,
  "driver_name": "Donnie Saunders",
  "amount": 500,
  "date": "03/21/2022",
  "payment_mode": "JazzCash"
}
```

Limitations (HINT):

- Package will start from 1st of that month, no matter when you join in that month. E-g: Zubair joins on 1st of oct and Ali joins on 20th of Oct, first month will expire on on 31-Oct 11:59 for both drivers
- All the dates will be in 2022. You can just check the month part instead of going for a full date comparison.
- Date format is MM/DD/YYYY

Input Format

The input will be read from a file. The filename/path will be passed to your program as the first command-line argument. First line of input will be an integer T representing the number of test cases. Each test case contains a comma separated driver ID and Joining date of the driver.

Output Format

Output will be the amount that needs to be paid by Driver or by Company. Amount driver needs to pay will be negative. Format your output to fixed 1 decimal places.

Sample Test Case Input:

2
10017, 04/01/2022
10011, 01/01/2022

Sample Test Case Output:

9634.4
-825.0

Car Manufacturing Design Problem

It is the future. Most of our jobs have been replaced by robots. Countries are run by sentient AIs and infrastructure, and agriculture is handled completely by robots. The only jobs that humans can get are as programmers. And all of the code they write is for the robots to do their jobs better. Car manufacturers were the last ones to switch completely to robots. That is why they're still figuring out a lot of things. They want to automate the assembly of a car.

You have been tasked with accomplishing this. The codebase is partially implemented. **Your job is to design the car assembly module**. Your solution needs to be object-oriented, which means you can write classes/interfaces/abstract classes or whichever practice seems right to you. Once you're done designing the classes, the architecture should at least provide solutions to the followings:

- Assemble cars
- Provide details on the Make/Model/Variant of each car
- Keep inventory of assembled cars
- List the features of a car

- Bonus: Given a feature, output the number of cars that have that feature

To assemble a car, the following are required, you can increase the scope if you want however the following must:

- Make/Model (Mercedes/S Class, Dodge/Charger, etc.)
- Body Type(Sedan, SUV, Hatchback)
- Engine Type(V1 to V8)
- Engine CC (1000, 1300, 1500, etc)
- Paint Color (Could be any color)
- Transmission (Auto, Manual)
- Features(Cruise Control, Heated Seats, Self Driving, etc. Each car can have multiple features)

You can take any creative liberties that you can think of, but you also need to ensure that the criteria defined above are met. Moreover, if you need to make any assumptions, please make sure to clearly mention them.

About this question

- This is an open ended question that will evaluate your OOP, data structures and general design skills.
- You can write your assumptions in your code comments
- There are no test cases for this program. So, no need to read anything from a file.

Your Answer

- Provide a running code with classes and functions
- Provide some test code to mimic different scenarios defined above
- Your code MUST be executable, otherwise it won't be submitted
- No need to save any data to the database or file system. You can hard code test data if needed.
- Final code should be properly commented and readable

MBS The Line

Mohammed bin Salman Al Saud (MBS) is the current crown Prince of Saudi Arabia. He is a visionary leader determined to reduce dependency of its regional economy on export of black gold(oil). He is planning to launch a new tourist attraction named "The Line". *"Often referred to as a 'groundscraper,' the singular structure will stretch 170 kilometers — over one hundred miles — to house a city of nine million people.The project is reinventing urbanism, seeking to remove the pollution, noise, and sprawl of city living in favor of an ultra-efficient utopia. To put the scale of THE LINE into perspective, it would take over an hour — perhaps nearly two hours — to drive that length by car. The team at NEOM claims that with its system of high-speed transport, one could reach end-to-end in no more than twenty minutes. Tall and narrow, the city will be 500 meters above sea level and 200 meters wide. (1,640 feet high and 650 feet wide). That's taller still than the 102-story Empire State Building and its spire."*

You are working on designing a simulator to determine the damage in case of natural disaster. You are provided the data of the cityscape in 2d array format. Each column (I) represents a building in a city. Each cell represents the weight **(W)** of a specific floor on that building.

Simulation Rules

- Your system will run the simulation X number of times (T(X), T(X-1) … T(1)). Simulation will start from the first row of input data.
- While running the simulation, floor (F) will merge with the floor below only if the total weight on all the floors above is greater than it.
- For any given building cell it will continue to merge until it reaches the limit of merges allowed (**Y**) per step. On each merge, the new weight of the cell will be the sum of floors involved in destruction.
- Single merge is defined as all the weights that get summed with respect to a single floor. (See example for further detail)

Final Result

- Your system will output the height of each building after Xth step.

Input

The input will be read from a file. The filename/path is passed to your program as the first command-line argument. The file will contain:

- Number of steps **(X)** to simulate.
- Allowed consecutive vertical merges **(Y),** for each step per building.
- Number of rows **(R)** to follow:
  - 2D array of data (i.e 1st building has floors at weigh 1,2,4 for 1st, 2nd and 3rd floor respectively)

Output

- Comma separated height of each building after running simulation for X steps

Constraints

- $1 < X <= R$
- $0 < Y <= 10^{\wedge 5}$
- $1 < R < 10^{\wedge 5}$
- **$0 <= W < 10^{\wedge 10}$**

Example Input

2
1
3
4,4,1
2,1,1
1,1,1

Solution

Legend

	Cells destructed
	Cell unaffected
	Active Floor

Initial cityscape			After destruction on 3rd floor			After destruction on 2nd floor		
4	4	1	0	0	1	0	0	0
2	1	1	6	5	1	0	0	0
1	1	1	1	1	1	7	6	3

	First Column: 4>2 hence it will merge into 6. (4+2) is a single merge. 6 > 1 but it wouldn't merge as the limit per step is 1 **Second Column:** 4>1 hence it will merge into 5. (4+1) is a single merge. 5 > 1, but it wouldn't merge as the limit per step is 1 **Third Column:** No change since (1 > 1) is false	

Output

1,1,1

Example Input

3
1
3
4,0,0
2,1,0
1,1,1

Solution

Legend

	Cells destructed
	Cell unaffected
	Active Floor

Initial cityscape	After destruction on 3rd floor	After destruction on 2nd floor	After destruction on 1st floor
4 0 0 2 1 0 1 1 1	0 0 0 6 1 0 1 1 1	0 0 0 0 1 0 7 1 1	0 0 0 0 1 0 7 1 1
	First Column: 4>2 hence it will merge into 6. (4+2) is a single merge. 6 > 1 but it wouldn't merge as the limit per step is 1		

Output

1,2,1

Example Input

2
2
4
2,7,0
2,1,0
1,1,1
1,1,1

Solution

Legend

	Cells destructed
	Cell unaffected
	Active Floor

Initial cityscape	After destruction on 4th floor	After destruction on 3rd floor
2 7 0 2 1 0 1 1 1 1 1 1	2 0 0 2 0 0 1 9 1 1 1 1	0 0 0 0 0 0 0 9 1 6 1 1
	Second Column: (7>1) therefore it will merge to a new value of 8. This is the first merge. (8>1) therefore it will merge again. This is the second merge. Only 2 merges are allowed as per input.	

Output

1,2,2