



[CS 11] Prac 9d – Shaman III: Shaman King

Problem Statement

Endugu is a shaman who resides in a dungeon hidden deep in the jungle. Unbeknownst to most, he has a day job of being a jeepney operator.

Today is a good day—there are so many passengers! They are in line waiting for a jeepney.

Endugu's task is to split the passenger queue into several groups, where each group will ride together in a jeepney.

Some jeepneys can accommodate up to 9 passengers on both sides (that's why Endugu often shouts "shaman yan!"), so they can accommodate up to 20 passengers overall (if we count the seats in front). However, some jeepneys may be able to accommodate less.

Given the sequence of passengers, as well as the number of passengers the jeepneys can accommodate in order, please split them up into their corresponding jeepney groups. It is okay for the last group to have less than the maximum passengers the jeepney can accommodate, if there aren't enough to reach it. However, please keep the order of the passengers the same.

It is guaranteed that there are enough seats to accommodate all passengers.

Every group should contain at least one passenger.

[Submit solution](#)

[CS 11]

Practice 9

[My submissions](#)

✓ Points: 200 (partial)

⌚ Time limit: 4.0s

GMEMORY limit: 1G

☒ Author:

kvatienza (Kevin Atienza)

➤ Problem type

☒ Allowed languages

NONE, py3

Task Details

Your task is to implement a function called `jeepney_groups`. This function has two parameters:

- `passengers` — an iterable of `str`s denoting the sequence of passenger names in their order in the queue.
- `jeepney_counts` — an iterable of `int`s denoting the number of passengers each jeepney can accommodate, in order.

The function must return a *generator* of `tuple`s. Each `tuple` must consist of `str`s denoting a group of passengers.

Note that your generator must be **as lazy as possible**. It should yield each resulting next element as soon as it has enough information, and it should produce these results while advancing the input generators for as little as possible.

Restrictions

(See 9a for more restrictions)

For this problem in particular:

- Recursion is **disallowed**. (The recursion limit has been greatly reduced.)
- The source code limit is 2000.

Example Calls

Example 1 Function Call

```
[*jeepney_groups(  
    iter(('eiko', 'dagger', 'freya', 'amarant', 'vivi',  
    'steiner', 'zidane', 'quina')),  
    iter((3, 4, 1)),  
)]
```

Copy

Example 1 Return Value

```
[  
    ('eiko', 'dagger', 'freya'),  
    ('amarant', 'vivi', 'steiner', 'zidane'),  
    ('quina',),  
]
```

Copy

Example 2 Function Call

```
[*jeepney_groups(  
    ('eiko', 'dagger', 'freya', 'amarant', 'vivi', 'steiner',  
    'zidane', 'quina'),  
    (3, 4, 20),  
)]
```

Copy

Example 2 Return Value

```
[  
    ('eiko', 'dagger', 'freya'),  
    ('amarant', 'vivi', 'steiner', 'zidane'),  
    ('quina',),  
]
```

Copy

Constraints

- The function `jeepney_groups` will be called at most 100 times.
- At most 80 elements will be consumed from the returned generator.
- Each jeepney can accommodate between 1 and 20 passengers.
- Each name is nonempty and consists of up to 8 lowercase English letters.
- It is guaranteed that there are enough seats to accommodate all passengers.

Scoring

- You get 125 ❤ points if you solve all test cases where:
 - all jeepneys will be fully occupied.
- You get 75 ❤ points if you solve all test cases.

Clarifications

[Report an issue](#)

No clarifications have been made at this time.