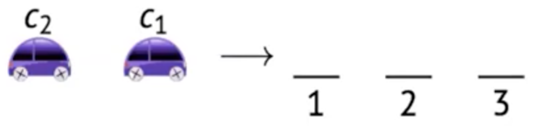
MTH 3300
Fall 2022

Programming Problem 4 parking.py

Directions: Download the template file I have provided on Blackboard. Then open Spyder, load these template files, and write the following program. Submit your source code via Gradescope in `.py` format. READ THE INSTRUCTIONS on submitting your work in the Course Documents section of Blackboard.

Be sure to read the SPECIFICATIONS carefully! And write comments!



[1]

A one-way street has three parking spots, numbered 1, 2, and 3. Two cars enter the street, one at a time. Each car has a preferred spot and will park according to the following rules:

- Car 1 always gets to park first. Car 2 will go next.
- If the car's preferred spot is available, it will attempt to park there.
- If the car's preferred spot has been taken, it will attempt to park in the next available spot it sees. (No backing up on the one-way street! For example, if a car's preference is spot 2 and spot 2 is taken, it will attempt to park in spot 3. If a car's first preference is spot 3, and spot 3 has been taken, the car will not park.)

Write a program asking for two cars' preferred spots and give the resulting parking configuration, assuming the vehicles follow the rules above. If one of the cars cannot park, your program should output this information (as specified in the sample run below).

A sample run might look like this (where the user enters the first `3` and the first `1`, and everything else is output by the program):

```
Car 1 preferred spot: 3
Car 2 preferred spot: 1
Car 1 parks in spot 3
Car 2 parks in spot 1
```

(since car 1's preferred spot is not taken and then car 2's preferred spot is not taken), or (where the user enters the first and second 3's):

```
Car 1 preferred spot: 3
Car 2 preferred spot: 3
Car 1 parks in spot 3
Car 2 cannot park
```

(since car 1's preferred spot is not taken, car 2 has no *next* available place to park).

Hint: there are nine possible sequences of inputs to this program. One way to do this problem is by writing one case for each sequence – a little tedious, but not too bad.

Note: This problem has six visible test cases on Gradescope. You will see if your code passes these 6 test cases. In addition, it has three invisible test cases, which you will not be able to see until after grades are released (you will not see if your code passed or did not pass these). So make sure you run additional tests on your own to cover all the possible inputs described! (You will need to determine by hand what the correct output should be in your test cases.)

Specifications: your program must

- ask the user to input the preferences of each of the two cars: `1`, `2`, or `3`. Use the input prompt format as specified in the sample runs above.
- print out the parking configuration, given that the cars will park according to the rules specified above. (Use the output format as in the sample runs above.)
- NOT use lists or tuples (or other structures we have not yet discussed)!

[1]from Pamela Harris's talk https://www.youtube.com/watch?v=WRHQRVXljR8

Interested in knowing more about what this parking business is about?

The idea for this assignment was inspired by a talk at a recent virtual conference I attended (ask me about it!). Believe it or not, the idea of *parking functions* is an area of active mathematics research. (Things get a lot more complicated when we generalize the problem to have n cars and n spots, and there are other generalizations too!)

I encourage you to read more in the articles linked below or watch the talk from the conference also linked below.

`http://www.girlsangle.org/page/bulletin-archive/GABv14n02E.pdf`

`http://www.girlsangle.org/page/bulletin-archive/GABv14n03E.pdf`

`https://www.youtube.com/watch?v=WRHQRVXljR8`

Did you get through parking.py quickly and want more fun things to program?

Extra Challenge: Program the same problem, but now with 3 cars rather than 2 (still 3 parking spots). To make it a bit simpler, in any case where ALL 3 of the cars can't park, your program should output the following:

```
not possible for all vehicles to park
```

(For example, it would not be possible for all cars to park if they all had spot 2 as their preference.) You'll likely have to do some work by hand before programming!

Please submit it to Programming Problem 3b, parkingchallenge.py, on Gradescope! If your code works correctly (passes all test cases), is commented on explaining your logic, and follows all specifications, you will earn a 10% bonus (11/10) on this problem.