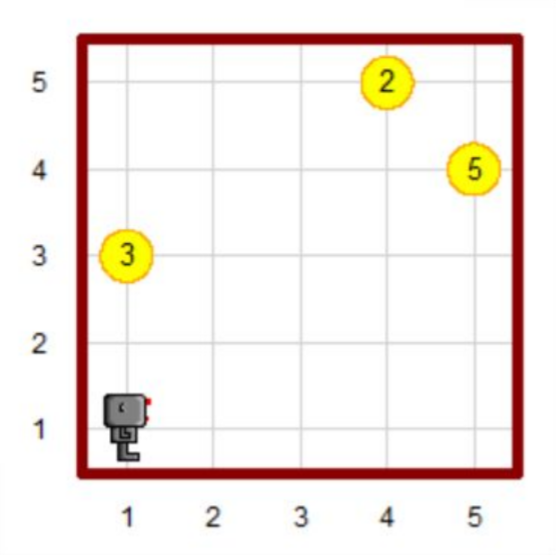
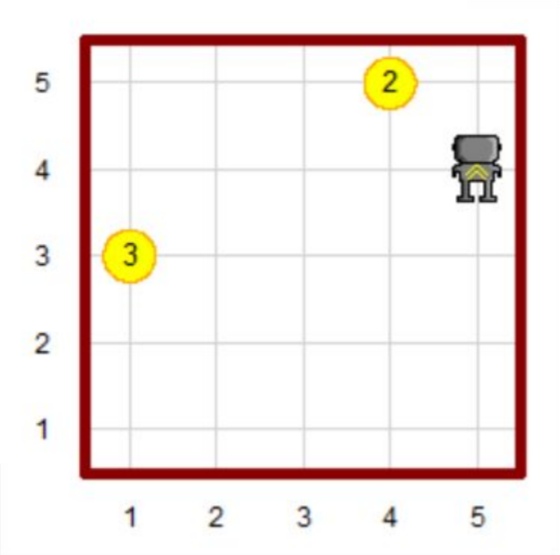
1. **Christmas Present for Hubo**

It’s Christmas! The developer of Hubo wants to celebrate the holiday by giving Hubo a Christmas Present. So, they have prepared many present boxes, each filled with different number of beepers, that Hubo can choose from. The developers tell Hubo that it can only choose ONE of the boxes as its Christmas gift. Hubo, being a greedy robot, wants to select the present box with the most beepers! Your job is to help Hubo so that it can select and pick up the present box with the most beepers by programming it. Below is an example world where the developers have positioned the present boxes (stacks of beepers).

**Example:**

**Task:**

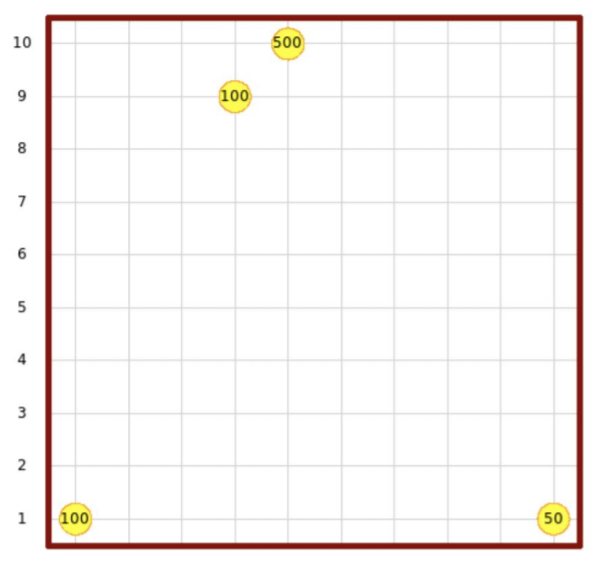
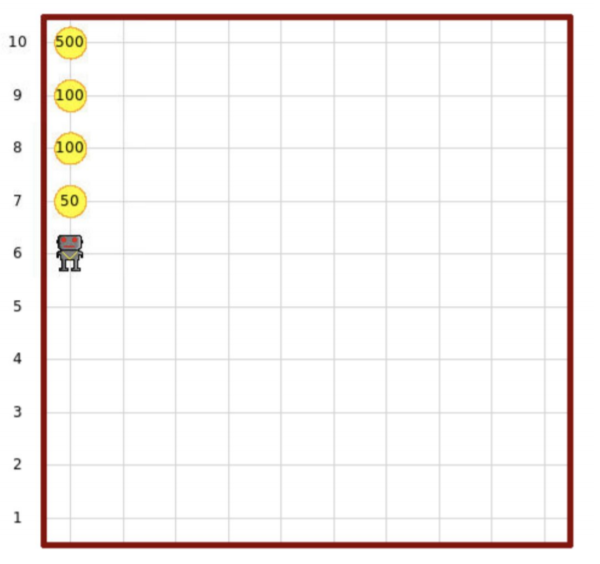
Given a world that has more than one beeper dropped on the ground, write a code that sends a robot to the position where there are the maximum number of beepers and pick up all beepers at that position.

**Assumptions:**

1. The world size is n x m, where both n and m are integers between 1 and 10.
2. No two stack will have the same number of beepers.
3. There will be at least one stack in each world.
4. The world will not have walls in the inside (rectangular shaped wall with no wall inside).
5. **Arranging Coins:**

Someone has spilled some coins on the ground! The developer of Hubo decided to let Hubo pick up all coins from the ground and arrange the coins in a certain way. Your job is to write a program that makes Hubo pick up all the coins (stacks of beepers) and arrange the coins in a descending order from the top street in the leftmost avenue.

**Example:**

**Task:**

Given a world, write a code that makes Hubo rearrange the coins in the world into the left-most column in a descending order from the top row. (Note, Hubo may not possess more than 1 coin at the same time)

**Assumptions:**

1. The world size is n x m, where both n and m are integers larger than or equal to 1.
2. The number of beeper stacks is not larger than the height of the world.
3. **Top 200 Times World University Rankings**

The Times Higher Education has been providing trusted performance data on universities for students and their families, university academics, university leaders, governments and industry, since 2004.

In this problem, you will be given a csv file containing data about the top 200 universities from 2011 to 2016. Each line of the csv file will have the following format:

<World Rank>, <University Name>, <Country>, <Year>

**Task 1)** Make a dictionary top\_200\_university\_dict using the csv file given with the following requirements:

* The keys of the dictionary are the names of the university.
* The value of each key should be a dictionary which has keys ‘Country’, ‘Ranking’ and ‘Year’.

**Example:**

top\_200\_university\_dict = {

...

'Korea Advanced Institute of Science and Technology (KAIST)': {

'Country':'South Korea’

'Ranking':[79, 94, 68, 56, 52, 148],

'Year':[2011,2012,2013,2014,2015,2016]

},

...

'Yonsei University': {

'Country':'South Korea’

'Ranking':[190, 183, 190],

'Year':[2011,2013,2014]

},

...

}

**Task 2**) Make a function university\_list(dictionary) that takes the dictionary made in task 1 as input and outputs a **list** of all the universities in the dictionary. Also, make a function university\_set(dictionary) that takes the dictionary made in task 1 as input and outputs a **set** of all the universities in the dictionary.

* The university list should not contain duplicate items.
* Please do not use the set(list) function to make a list into a set.

**Task 3**) Make a function check\_ranking(dictionary, name, year) that takes a dictionary, university name, and year as inputs and outputs the ranking of the given university in the given year. This function should be able to handle exceptional cases (given name not in dict, no result for given year, etc…).

**Example:**

>>> dict = top\_200\_university\_dict

>>> check\_ranking(dict, 'wrong univ name', 2012)

No matching university name.

>>> check\_ranking(dict, 'Harvard University', 2000)

No result for given year.

>>> check\_ranking(dict, 'Harvard University', 2012)

6

**Task 4**) Make a function average\_ranking(dictionary, name) that takes a dictionary, and university name and returns the average ranking of the given university from 2011 to 2016. If the university is out of ranking in a certain year, assume the ranking is 200. Again, this function should handle exceptional cases.

* Which university has the highest average ranking?
* Which university has the lowest average ranking?

**Example:**

>>> dict = top\_200\_university\_dict

>>> average\_ranking(dict, 'wrong univ name')

No matching university name.

>>> average\_ranking(dict, 'Harvard University')

2.8333333333333335

**Submission**

Please submit your code by emailing them to **BOTH**:

* [steve-kim@kaist.ac.kr](mailto:steve-kim@kaist.ac.kr)
* [dkcjd2000@kaist.ac.kr](mailto:dkcjd2000@kaist.ac.kr)