CS 245 - Lab 9 Fall, 2021

Lab 9 - Among Us

The goal of this assignment is to demonstrate your mastery of ~~*Among Us*~~ **graph**. Download your code here.

Background

Attention please. There might be an imposter among us. The spaceship has `n` crewmates labeled from 1 to n.

If the imposter exists, then

1. The imposter suspects nobody.
2. Everyone can suspect one other player.
3. It is guaranteed that exactly one player will satisfy 1 and 2.

You are given an array `suspect` where `suspect[i]=[a_i,b_i]` representing that the person labeled `a_i` suspect the person labeled `b_i` .
Find the imposter if there exists and return the label of the imposter, otherwise return -1.

Constraints:

- `1 <= n <= 1000`
- $0 <= suspect.length <= 10^4$
- `suspect[i].length == 2`
- `All the pairs of suspect are unique.`
- `ai != bi`
- `1 <= ai, bi <= n`

Examples:

```
Input: n = 2, suspect = [[1,2]]
Output: 2


Input: n = 3, suspect = [[1,3],[2,3]]
Output: 3

Input: n = 3, suspect = [[1,3],[2,3],[3,2]]
Output: -1

Input: n = 3, suspect = [[1,2],[2,3]]
Output: -1

Input: n = 4, suspect = [[1,3],[1,4],[2,3],[2,4],[4,3]]
Output: 3
...
```

Requirements

- You must submit a single Java file containing the class named **AmongUs**. You class must contain the function named "`findImposter`" with the following header:

```
public int findImposter(int n, int[][] suspect)
```

Submission

- Download the source code for your implementation from Github Classroom. If you don't have a Github account, create one via github.com > Sign up.
- Select your name from the list of names available, which will link your GitHub ID. If you do not see your name on the list, speak with the instructor or one of the teaching assistants.
- Submit the repository link on Canvas. You may also add any comments in a README file (text, PDF or Word document) to help the grader understand or execute your implementation.

Grading

Your grade for this assignment will be determined as follows:

- 75% = Implementation: your class implementations must run successfully with the source files and data provided. It must produce the expected results, a sample of which appears in the Implementation section above. Any deviation from the expected results results in 0 credit for implementation.
- 15% = Decomposition: in the eyes of the grader, the degree to which your solution follows the suggestions above or otherwise represents a reasonable object-oriented and procedural decomposition to this problem.
- 10% = Style: your code must be readable to the point of being self-documenting; in other words, it must have consistent comments describing the purpose of each class and the purpose of each function within a class. Names for variables and functions must be descriptive, and any code which is not straightforward or is in any way difficult to understand must be described with comments.