Assignment 4: 20 marks

No minor-clarifications will be provided. Assignments are also meant to improve your decision-making. Any major change (unlikely) will be clarified from our side on Piazza. But take your call on implementation level details.

Write a Java program **`assignment4.java`**, that takes three command line arguments: two csv files (samples given below), and a function name. Based on the function name, your program output changes.

Marvel characters graph

nodes.csv 327 nodes: Each node represents a character.
edges.csv 9891 edges (with weights): Edges between nodes/characters represents that the characters/nodes have appeared together. Edge-weights are proportional to the number of co-occurrences.
Note: Our test graphs will be different from these. Hard-coding output values in code for these particular graphs will be caught easily and award you a 0 in the whole assignment component.

Implement the information in the two csv files as a graph and write functions to output the following:

(a) Function: average. Should print the average number of characters each Marvel character is associated with, as a float upto two decimal places. [2 marks].
Sample random output:

```
$java assignment4 nodes.csv edges.csv average
7.43
```

(b) Function: rank. Should print a sorted list of all characters, with comma as delimiter (only comma, as delimiter and no space). Sorting should be in descending order of co-occurrence with other characters. That is, characters with more co-occurrences appear before. If there is a tie between characters based on co-occurrence count, then the order should be descending based on lexicographic order of the character strings.
You can implement any O(nlogn) sorting algorithm. The test input file will be a huge graph and we will run the evaluation cases with timeout. If you implement O(n^2) sorting instead of O(nlogn) then the cases will timeout, and you will lose marks. Also don't use Java inbuilt sorting, but implement on your own. We will check your imports against allowed_imports.txt. [8 marks]
Sample random output:

```
$java assignment4 nodes.csv edges.csv rank
Yogish,Riju,Rahul
```

(c) Function: independent_storylines_dfs. Should implement DFS, then find independent storylines, that have no edge across them, using DFS. Print the characters in each independent storyline, as a separate line in the output.
The largest storyline (with maximum characters) should appear at the top, followed by the second largest and so on. Within each line, the character names should be delimited with comma, and lexicographically sorted in descending order. If two storylines have same number of characters, ties should be broken in lexicographically descending order of character names. [10 marks]
Sample random output:

```
$java assignment4 nodes.csv edges.csv independent_storylines_dfs
Riju,Rahul
Yogish
```