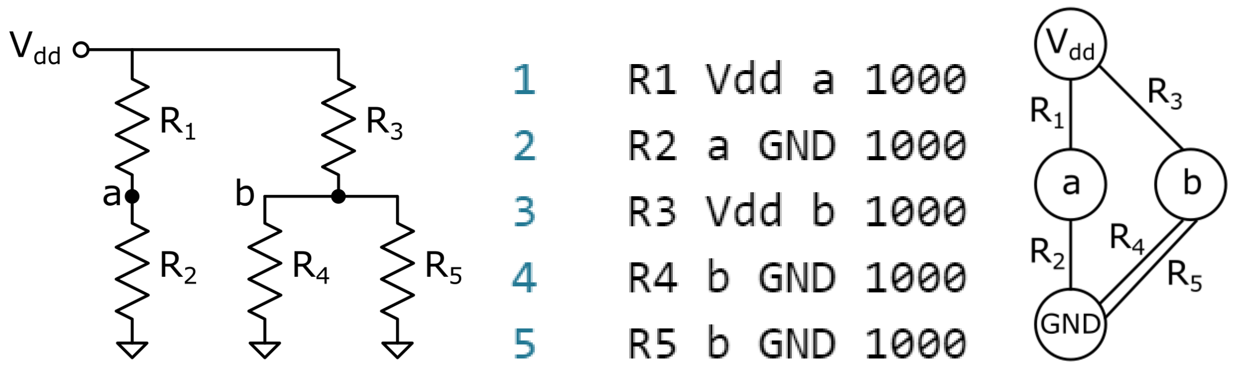
Software Project - Resistor Network Calculator

This project is to be done **individually**. Discussion among students is allowed although explicitly *copying each other's work will be considered an act of academic dishonesty*. Submissions for each milestone will be subjected to a similarity checker and those whose submissions are too similar will not be graded and may be elevated to a disciplinary case. Please **cite all sources used**, both offline and online, as a comment in your source code.

Problem Statement

Since an electronic circuit is the interconnection of several components, it might not be surprising that they can be represented as graphs which allow our computers to simulate the behavior of the circuit. Shown in Figure 1 is a sample resistor network with its corresponding netlist representation used in `ngspice` which is technically the edge list of a graph. Each node in the circuit is a vertex in the graph, while each resistor is an edge with a certain weight. In this software project we will attempt to create our own circuit solver, or at least a resistance calculator using graphs and the other data structures and algorithms you have encountered in this course.



```
R1 Vdd a 1000
R2 a GND 1000
R3 Vdd b 1000
R4 b GND 1000
R5 b GND 1000
```

Figure 1. A sample resistor network
(left) circuit diagram, (middle) `ngspice` netlist, and (right) graph representation.

Resistance can be thought of as the **cost** for an electron to travel from one node to another. That being the case, the total resistance between two nodes can be simply calculated as the total cost of the path between them. As an example in Figure 1, if we only consider R_1 and R_2, the total resistance between V_{dd} and GND is $R_1 + R_2$, or, in other words, their series combination.

However, if multiple paths (resistors) exist between nodes, these multiple paths would appear to be in parallel, an example of which is between node b and GND in Figure 1. To get the total resistance between the nodes, we can combine the conductances of each path (i.e. getting the reciprocal of the sum of their reciprocals).

Even though we are only dealing with resistors, our project can get overwhelming. To help, this project is divided into ***5 smaller subproblems or milestones*** each worth a percentage of your overall grade.

Milestone 1 - R You Connected? (15%)

Description

This first milestone focuses on properly parsing the input. To test this, your program should be able to properly store each input resistor and store them in a data structure and representation of your choice.
To verify this, your program should be able to identify which resistors are in series and which resistors are in parallel. In the context of a graph, two resistors can be considered in series if there are no branches in the path formed between the resistors. On the other hand, two resistors can be considered in parallel if they are connected to the same two nodes.

Input Format

The first line of the input consists of 2 integers N and Q which is the total number of resistors and total number of queries, respectively. The next N lines will consist of 3 strings and an integer R, each separated by a space. The first string is the resistor name, the second and third strings are the node names where the resistor terminals are connected, and R is its resistance value. Finally, there will be Q lines with two strings each which will each be the name of a resistor.

Output Format

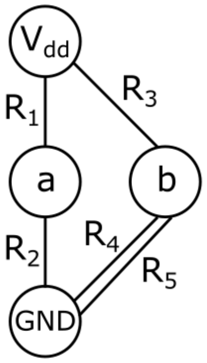
The output is expected to have Q lines which correspond to the answer to each query. If the resistors in the first query are in series, the first line of the output should be **`SERIES`**. If they are parallel, the expected output is **`PARALLEL`**. If they are neither in series or parallel, the expected output is **`NEITHER`**.

Constraints

- Resistors in the queries are guaranteed to exist.
- Your code needs to finish within 10s on HackerRank.
- $1 \leq N, Q \leq 1000$ and $0 < R \leq 10^9$

Sample Input and Output

Input 0	Output 0	Visualization

`5 4`
`R1 Vdd a 1000`
`R2 a GND 1000`
`R3 Vdd b 1000`
`R4 b GND 1000`
`R5 b GND 1000`
`R1 R2`
`R2 R3`
`R3 R4`
`R4 R5`
	`SERIES`
	`NEITHER`
	`NEITHER`
	`PARALLEL`
		 V_{dd}
		R_1
		R_3
		a
		b
		R_2
		R_4
		R_5
		GND

Milestone 2 - Put TogetheR (15%)

Description

This second milestone tests for two things. Your program must identify all resistors in series or in parallel with each other and calculate the equivalent resistance for each group of resistors. Recall that total resistance of resistors in series are added up while the total conductance of those in parallel are added up.

Input Format

The first line of the input consists of 1 integer N which is the total number of resistors. The next N lines will consist of 3 strings and an integer R each separated by a space. The first string is the resistor name, the second and third strings are the node names where the resistor terminals are connected, and R is its resistance value.

Output Format

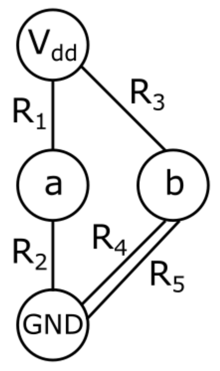
One line for each group of resistors in series or parallel. Each line starts with a list of resistors arranged in lexicographical order followed by their total resistance rounded off to the nearest integer using the built-in `int()` Python function. The order that the groups are printed should also be in lexicographic order to ensure proper checking.

Constraints

- Your code needs to finish within 10s on HackerRank.
- Be as precise as possible in the computation to avoid rounding errors in the final answer.
- $1 \leq N \leq 1000$ and $0 < R \leq 10^9$

Sample Input and Output

Input 0	Output 0	Visualization

`5`
`R1 Vdd a 1000`
`R2 a GND 1000`
`R3 Vdd b 1000`
`R4 b GND 1000`
`R5 b GND 1000`
	`[R1, R2] 2000`
	`[R4, R5] 500`
		 V_{dd}
		R_1
		R_3
		a
		b
		R_2
		R_4
		R_5
		GND

Milestone 3 - One LineR (15%)

Description

From milestone two, you should have a pretty good idea on how to reduce the number of resistors in the graph while maintaining the equivalent resistances between nodes. Much like how we would solve for the resistance by hand, we can redraw the circuit with the equivalent resistances between each node where there is a group of resistors in series and in parallel. You can then find these equivalent resistances in series or parallel with the other resistors and simplify the circuit further. For milestone 3, find the equivalent resistance between two nodes assuming that the resistor network can be simplified by only using multiple iterations of series and parallel combinations.

Input Format

The first line of the input consists of 1 integer N which is the total number of resistors. The next N lines will consist of 3 strings and an integer R each separated by a space. The first string is the resistor name, the second and third strings are the node names where the resistor terminals are connected, and R is its resistance value.

Output Format

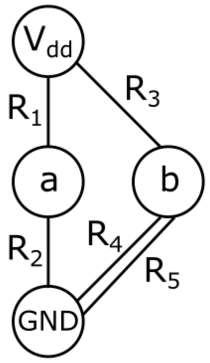
One line for the total equivalent resistance of the resistor network between the nodes **Vdd** and **GND** rounded off to the nearest integer.

Constraints

- Assume that the circuits given would not need Δ-to-Y or Y-to-Δ transformations.
- Your code needs to finish within 10s on HackerRank.
- Be as precise as possible in the computation to avoid rounding errors in the final answer.
- The nodes **Vdd** and **GND** are guaranteed to exist.
- $1 \leq N \leq 1000$ and $0 < R \leq 10^9$

Sample Input and Output

Input 0	Output 0	Visualization

`5`
`R1 Vdd a 1000`
`R2 a GND 1000`
`R3 Vdd b 1000`
`R4 b GND 1000`
`R5 b GND 1000`
	`857`	 V_{dd}
		R_1
		R_3
		a
		b
		R_2
		R_4
		R_5
		GND

Milestone 4 - Del(ta) me Y (15%)

Description

Our solution in Milestone 3 will probably not get the equivalent resistance of a network with a Y or Δ connection since these resistors are neither in series nor in parallel and cannot be simplified by adding resistance or conductances. For Milestone 4, the goal is to take account of this and create a program that can get the equivalent resistance between two nodes for any possible resistor network.

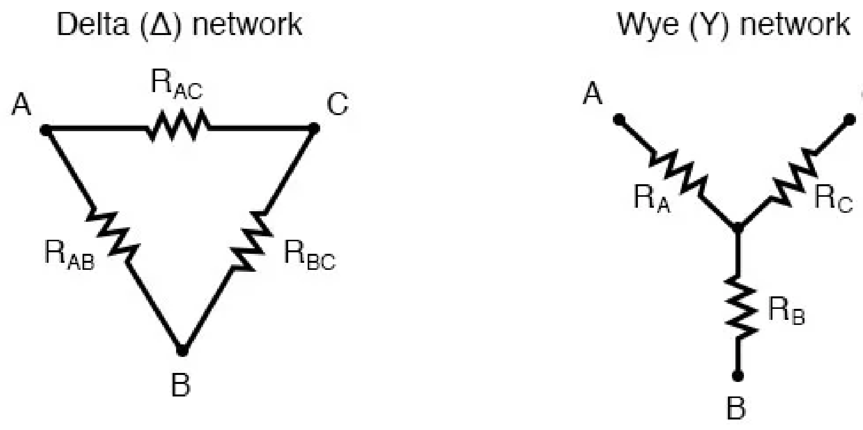


Figure 2. Delta and Wye Connected Resistors
(https://www.allaboutcircuits.com/textbook/direct-current/chpt-10/delta-y-and-y-conversions/)

Note that Y connected resistors can be converted to a Δ connected equivalent and vice-versa by using the equations below:

$$R_A = \frac{R_{AB}R_{AC}}{R_{AB} + R_{AC} + R_{BC}} \qquad R_{AB} = \frac{R_AR_B + R_AR_C + R_BR_C}{R_C}$$

$$R_B = \frac{R_{AB}R_{BC}}{R_{AB} + R_{AC} + R_{BC}} \qquad R_{BC} = \frac{R_AR_B + R_AR_C + R_BR_C}{R_A}$$

$$R_C = \frac{R_{AC}R_{BC}}{R_{AB} + R_{AC} + R_{BC}} \qquad R_{AC} = \frac{R_AR_B + R_AR_C + R_BR_C}{R_B}$$

(https://www.allaboutcircuits.com/textbook/direct-current/chpt-10/delta-y-and-y-conversions/)

After doing the conversion from one connection to another, you should now be able to find series or parallel connections in your circuit and continue with the simplification.

Input Format

The first line of the input consists of 1 integer N which is the total number of resistors. The next N lines will consist of 3 strings and an integer R each separated by a space. The first string is the resistor name, the second and third strings are the node names where the resistor terminals are connected, and R is its resistance value.

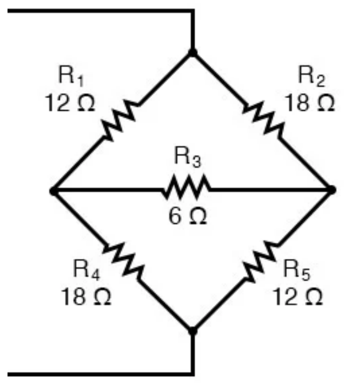
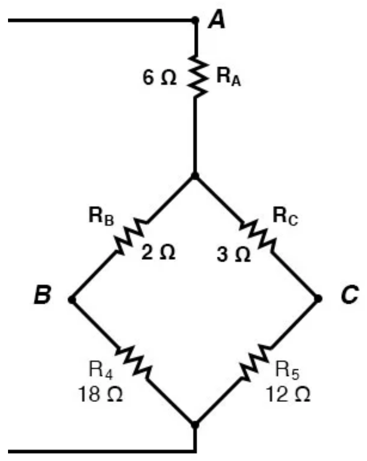
Output Format

One line for the total equivalent resistance of the resistor network between the nodes **Vdd** and **GND** rounded off to the nearest integer.

Constraints

- Your code needs to finish within 10s on HackerRank.
- The nodes **Vdd** and **GND** are guaranteed to exist.
- Be as precise as possible in the computation to avoid rounding errors in the final answer.
- $1 \le N \le 1000$ and $0 < R \le 10^9$

Sample Input and Output

Input 0	Output 0	Visualization	
5 R1 Vdd B 12 R2 Vdd C 18 R3 B C 6 R4 B GND 18 R5 C GND 12	15	 R1 12 Ω R2 18 Ω R3 6 Ω R4 18 Ω R5 12 Ω	 A 6 Ω RA RB 2 Ω 3 Ω RC B C R4 18 Ω R5 12 Ω
		(https://www.allaboutcircuits.com/textbook/direct-current/chpt-10/delta-y-and-y-conversions/)	

Milestone 5 - Documentation (40%)

Description

Since we only just considered resistors for our project, it will take quite a bit of time before it can compete with the likes of `ngspice`. As such, you must learn to properly document your approach to the problem so that others can build on it. A good documentation also highlights possible problems in the project and points for improvement.

Expected Content

For each milestone completed, answer the following:

- How does your final solution work? You can show snippets of your code and define the purpose of each function or line. You can also add a flowchart to visually show the step-by-step procedure.
- What was your starting point? Did you use code you made previously?
- From your starting point, what did you need to change to achieve the functionality required?
- Were there specific test cases you found to be very tricky? How were you able to address them?

For each milestone attempted but fails a few test cases, answer the following:

- How does your solution currently work?
- Which test cases were not satisfied by your solution? For hidden test cases,
  you may provide your own test cases which you know your code does not solve correctly.
- How does your result for these test cases compare with the expected results?
- What would you infer to be the cause of these mistakes and what would you do if given more time?

For each milestone you have not attempted, answer the following:

- Can you think of a possible solution for this milestone? It could help if you could reference
  your solutions to previous milestones and enumerate the changes you would need to do.
- What do you think would be the biggest challenge or step to implement in your proposed solution?
- How much longer do you think it would take to complete this milestone?

For each milestone whether completed or not, answer the following additional questions:

- What are the data structures used in your solution? Were these among the common ones we discussed in class or maybe a custom data structure based on one or a combination of these?
- What are the time and space complexities of your solutions? Do you think these could be improved?

Output Format

You may submit a scanned handwritten report, typewritten report, or a recorded presentation as long as it is understandable and covers the specified information above. Your report should be a single PDF if it is written and an MP4 file if it is recorded. There is no limit on the amount of words and length but please be as clear and concise as possible.

Submission and Grading

HackerRank Challenges will be set up for you to test your code for milestones 1, 2, 3, and 4. But similar to weekly exercises, the final submissions will be done through UVLe. A ZIP file containing the final versions of your source code and report for milestone 5 is expected. Even if your code is not able to pass all the test cases for a milestone, please do still submit it for partial points.

The *suggested* filename is "**`<surname>_<nickname>_<student_no>_SP_MS<N>.<ext>`**" where **`N`** is the milestone number and **`<ext>`** is the corresponding file extension (e.g. "**`py`**" for Python source codes and "**`pdf`**" for PDFs). Submissions for each milestone will be subjected to a similarity checker and those whose submissions are too similar will not be graded and may be elevated to a disciplinary case. Please **cite all sources used**, both offline and online, as a comment in your source code.

Each milestone can be started independently but since succeeding milestones supposedly build on the previous milestones, it is suggested that you do these sequentially. The following is a breakdown of how each milestone will be graded:

Milestone	**Percent Total**	**Breakdown**

Milestone 1 -
R You Connected?
	`15%`	`3%  - Sample Test Cases`
		`12% - Hidden Test Cases`

Milestone 2 -
Put TogetheR
	`15%`	`3%  - Sample Test Cases`
		`12% - Hidden Test Cases`

Milestone 3 -
One LineR
	`15%`	`3%  - Sample Test Cases`
		`12% - Hidden Test Cases`

Milestone 4 -
Del(ta) me Y
	`15%`	`3%  - Sample Test Cases`
		`12% - Hidden Test Cases`

Milestone 5 -
Documentation
	`40%`	`10% - For each milestone covered.`
		`(MS1, MS2, MS3, MS4)`

The halfway point soft deadline is set to 11:00 PM on **June 2, 2022**
and the hard deadline is set to 11:00 PM on June 16, 2022.

General Hints

- How you store data can limit what you can do with the data and may lead to less efficient code.
- You are free to use any of the data structures discussed, not just graphs, like maps, stacks, and disjoint sets.
- You may find milestones down the line may be easier for the specific approach you are going for.
  You may skip to solving that higher milestone then work your way back to previous ones.