Software Project

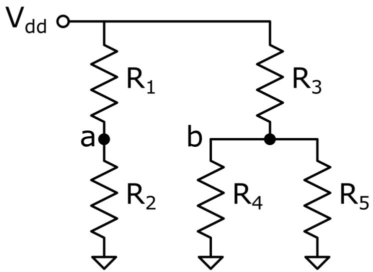
Resistor Network Calculator

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

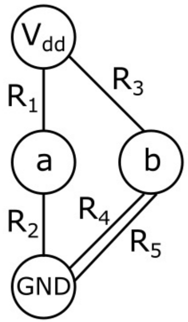
- Electronic circuits can be represented as graphs for simulation and analysis
- Each component in a circuit is a vertex in the graph, and connections are represented as edges
- Resistance can be seen as the cost for electron flow between nodes
- Total resistance between two nodes is the sum of resistors along the path connecting them
- Multiple paths between nodes can be treated as parallel resistors



circuit diagram

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

ngspice netlist



graph representation

Problem Statement

- 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.

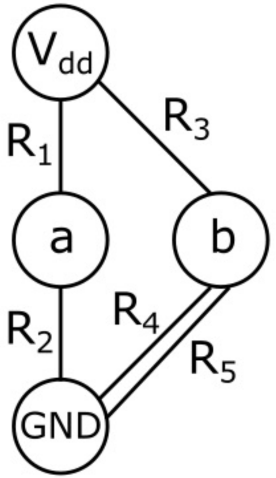
Milestone 1

- Parsing input and storing resistors
  - Program should store each input resistor in a chosen data structure
- Resistor identification:
  - Series: No branches in the path between the resistors
  - Parallel: Connected to the same two nodes
- Input format:
  - First line: Two integers N (total number of resistors) and Q (total number of queries)
  - Next N lines: Resistor information with 3 strings (resistor name, node names) and an integer R (resistance value)
  - Next Q lines: Two strings, names of resistors for each query

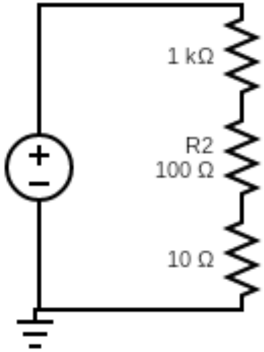
Milestone 1

- Output format:
  - Q lines corresponding to each query
  - SERIES: If resistors are in series
  - PARALLEL: If resistors are in parallel
  - NEITHER: If resistors are neither in series nor parallel
- Constraints:
  - Resistors guaranteed to exist in queries
  - Code execution within 10 seconds on HackerRank
  - $1 \leq N, Q \leq 1000$
  - $0 < R \leq 10^9$

Milestone 1 - Sample 0

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	

Milestone 1 - Sample 1

Input 1	Output 1	Visualization
3 6 R1 Vdd a 1000 R2 a b 100 R3 b GND 10 R1 R2 R1 R3 R2 R3 R2 R1 R3 R1 R3 R2	SERIES SERIES SERIES SERIES SERIES SERIES	

Milestone 1 - Sample 2

Input 2	Output 2	Visualization
3 6 R1 Vdd GND 1000 R2 Vdd GND 500 R3 Vdd GND 200 R1 R2 R1 R3 R2 R3 R2 R1 R3 R1 R3 R2	PARALLEL PARALLEL PARALLEL PARALLEL PARALLEL PARALLEL	

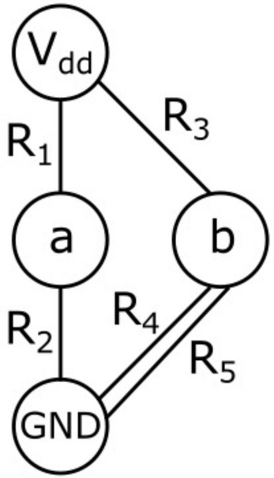
Milestone 2

- Identifying resistors in series or parallel and calculating equivalent resistance
- Identification:
  - Series: Resistors connected in a linear path
  - Parallel: Resistors connected to the same two nodes
- Calculation:
  - Total resistance in series: Sum of individual resistances
  - Total conductance in parallel: Sum of individual conductances
- Input format:
  - First line: One integer N (total number of resistors)
  - Next N lines: Resistor information with 3 strings (resistor name, node names) and an integer R (resistance value)

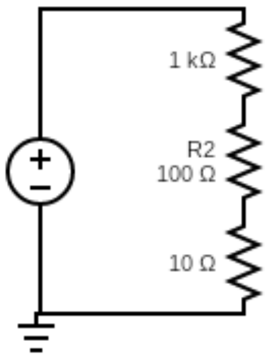
Milestone 2

- Output format:
  - One line for each group of resistors in series or parallel
  - Line format: List of resistors in lexicographical order, followed by their total resistance (rounded to the nearest integer using `int()` function)
  - Group order: Lexicographic order to ensure proper checking
- Constraints:
  - Code execution within 10 seconds on HackerRank
  - Be precise in computations to avoid rounding errors
  - $1 \leq N \leq 1000$
  - $0 < R \leq 10^9$

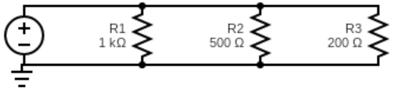
Milestone 2 - Sample 0

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	

Milestone 2 - Sample 1

Input 1	Output 1	Visualization
3 R1 Vdd a 1000 R2 a b 100 R3 b GND 10	[R1, R2, R3] 1110	

Milestone 2 - Sample 2

Input 2	Output 2	Visualization
3 R1 Vdd GND 1000 R2 Vdd GND 500 R3 Vdd GND 200	[R1, R2, R3] 125	

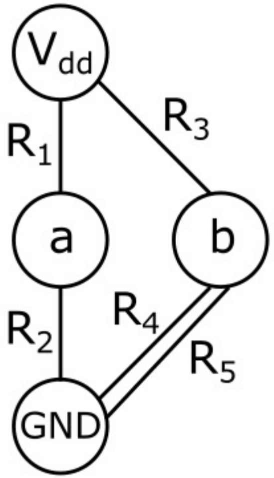
Milestone 3

- Finding the equivalent resistance between two nodes in a simplified resistor network
- Simplification approach:
  - Reduce resistors using series and parallel combinations
  - Redraw circuit with equivalent resistances between nodes
  - Continue simplifying the circuit by finding equivalent resistances in series or parallel
- Input format:
  - First line: One integer N (total number of resistors)
  - Next N lines: Resistor information with 3 strings (resistor name, node names) and an integer R (resistance value)

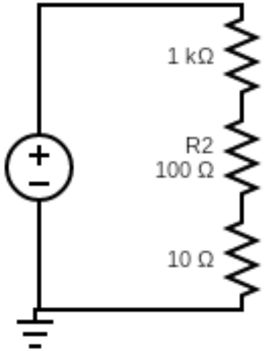
Milestone 3

- Output format:
  - One line: Total equivalent resistance between nodes Vdd and GND (rounded to the nearest integer)
- Constraints:
  - Assume no Δ-to-Y or Y-to-Δ transformations needed
  - Code execution within 10 seconds on HackerRank
  - Be precise in computations to avoid rounding errors
  - $1 \leq N \leq 1000$
  - $0 < R \leq 10^9$
  - Note: Nodes Vdd and GND are guaranteed to exist.

Milestone 3 - Sample 0

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	

Milestone 3 - Sample 1

Input 1	Output 1	Visualization
3 R1 Vdd a 1000 R2 a b 100 R3 b GND 10	1110	

Milestone 3 - Sample 2

Input 2	Output 2	Visualization

`3`
`R1 Vdd GND 1000`
`R2 Vdd GND 500`
`R3 Vdd GND 200`
	`125`	

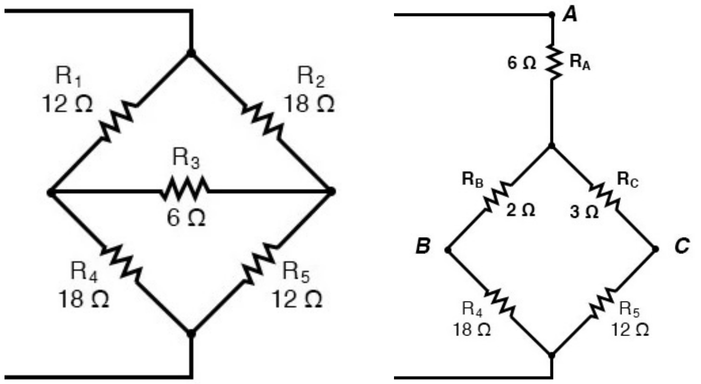
Milestone 4

- Finding the equivalent resistance between two nodes in a resistor network, accounting for Y and Δ connections
  - Consider Y and Δ connected resistors
  - Conversion between Y and Δ connections using specific equations
  - Simplify the circuit after the conversion by finding series or parallel connections
- Input format:
  - First line: One integer N (total number of resistors)
  - Next N lines: Resistor information with 3 strings (resistor name, node names) and an integer R (resistance value)

Milestone 4

- Output format:
  - One line: Total equivalent resistance between nodes Vdd and GND (rounded to the nearest integer)
- Constraints:
  - Code execution within 10 seconds on HackerRank
  - Be precise in computations to avoid rounding errors
  - $1 \leq N \leq 1000$
  - $0 < R \leq 10^9$
  - Note: Nodes Vdd and GND are guaranteed to exist.

Milestone 4 - Sample 0

<table>
<tr><th>Input 0</th><th>Output 0</th><th>Visualization</th></tr>
<tr><td>5
R1 Vdd B 12
R2 Vdd C 18
R3 B C 6
R4 B GND 18
R5 C GND 12</td><td>15</td><td>
</td></tr>
</table>

Milestone 5 - Documentation

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.

Milestone 5 - Documentation

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?

Milestone 5 - Documentation

Expected Content

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?

Milestone 5 - Documentation

Expected Content

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?

Milestone 5 - Documentation

Expected Content

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?

Milestone 5 - Documentation

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.

Submission and Grading

- 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.

Submission and Grading

- 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)`

Submission and Grading

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

Reminder

You are required to get at least 30% out of a maximum of a 60% from the Weekly Activities and **a total of at least 20% out of a maximum of 40% from the Long Exam and Software Project combined**.