Task 4: `CityMapping`

Peanut is going for a holiday to the city Silvermill, and therefore he needs a map of Silvermill. Unfortunately, his computer is broken, and so he cannot Google for the map.

Silvermill is a city consisting of N road junctions, numbered 1 to N, and $N-1$ roads, with each road i connecting two road junctions A_i and B_i in both directions. This road also takes W_i minutes to traverse in either direction. It is guranteed that the city is fully connected, meaning it is possible to travel from any road junction to any other road junction using the roads in the city. To avoid congestion, the governor of Silvermill has decided that there will be **no more than three roads connected to each junction** in the city.

Peanut needs to obtain a map of Silvermill. In other words, he needs to obtain (A_i, B_i, W_i) for $i = 1, \ldots, N-1$.

Peanut has hired a cartographer in Silvermill to accomplish this task. As the cartographer is not very skilled, he can only answer a very simple question: what is the minimum amount of time, in minutes, required to travel between road junctions X and Y? Peanut did not pay enough money to this cartographer, so he only answers Q such questions before he quits his job.

Peanut needs you to help him to design a list of questions so that he can map out Silvermill and enjoy his holiday. Can you help him?

Implementation Details

In this task, you are asked to implement the following function: (It is different from the normal task, which requires the program to read from standard input and output to standard output.)

```
void find_roads(int N, int Q, int A[], int B[], int W[])
```

This function will be called exactly once with two input parameters and three output parameters. The two input parameters are N and Q. N is the number of road junctions while Q is the maximum number of queries allowed. The three output parameters are A, B and W. You are required to determine the $N-1$ roads in the city and return them in the arrays A, B and W. The endpoints of the roads and the roads themselves can be in any order.

You are allowed to call the following grader function to complete the task:

```
long long get_distance(int X, int Y)
```

This function will return a single integer, the time taken to travel between road junctions X and Y, in minutes. If you call this function more than Q times or provide invalid road junction numbers as arguments, the program will terminate immediately and you will be given a *Wrong Answer* verdict.

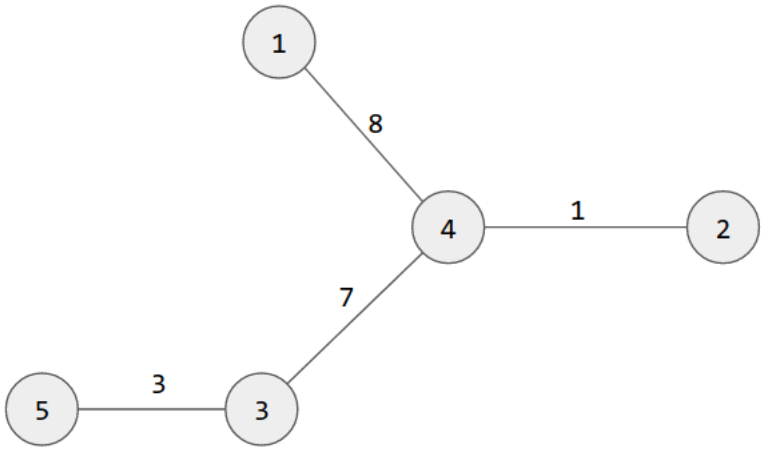

Figure 4: A city with 5 road junctions and 4 roads.

Consider the city map found in Figure 4. Suppose $Q = 500000$. Then, you are asked to implement the function:

```
find_roads(5, 500000, A[], B[], W[])
```

A possible interaction between your program *find_roads* and the grader functions *get_distance* could be as follows:

- `get_distance(5, 4) = 10`

  The *get_distance* function is asked to find the distance between road junctions 4 and 5. The answer is 10, as it takes 3 minutes to get from road junction 5 to 3, then another 7 minutes to get from road junction 3 to 4.

- `get_distance(2, 4) = 1`

  The *get_distance* function is asked to find the distance between road junctions 2 and 4. The answer is 1, as it takes 1 minute to directly travel from road junction 2 to 4.

- `get_distance(1, 3) = 15`

  The *get_distance* function is asked to find the distance between road junctions 1 and 3. The answer is 15, as it takes 8 minutes to get from road junction 1 to 4, then another 7 minutes to get from road junction 4 to 3.

- `get_distance(1, 2) = 9`

  The *get_distance* function is asked to find the distance between road junctions 1 and 2. The answer is 9, as it takes 8 minutes to get from road junction 1 to 4, then another 1 minutes to get from road junction 4 to 2.

At this point, the contestant's *find_roads* function decides it has enough information to deduce the map of the city. It sets $A = [3, 4, 4, 5]$, $B = [4, 1, 2, 3]$ and $W = [7, 8, 1, 3]$ and then terminates. This is a possible correct answer. The endpoints of the roads and the roads themselves can be provided in any order.

Subtasks

The maximum execution time on each instance is 1.0s. Your program will be tested on sets of input instances that satisfy the following restrictions:

- $2 \leq N \leq 1000$,
- $1 \leq A_i, B_i \leq N$,
- $1 \leq W_i \leq 10^9$.

Subtask	**Marks**	Q	Additional Constraints
1	9	$Q = 500000$	$W_i = 1$.
2	16	$Q = 500000$	No additional constraint.
3	13	$Q = 12000$	Each junction has at most **two roads** connected to it, $W_i = 1$.
4	19	$Q = 12000$	Each junction has at most **two roads** connected to it.
5	43	$Q = 25000$	No additional constraint. Please read *Scoring* section for further details.

Scoring

Subtask 5 is a special scoring subtask. Your score depends on the maximum number of queries q you make in any testcase.

- If $q > 25000$, you will score 0 points.
- If $12000 < q \leq 25000$, you will score $10 - 10 \times \frac{q-12000}{13000}$ points.

- If $6500 < q \leq 12000$, you will score $40 - 30 \times \frac{q-6500}{5500}$ points.
- If $q \leq 6500$, you will score 43 points.

Testing

You may download the grader file, *grader.cpp*, the header file, *citymapping.h*, a solution template, *citymapping.cpp*, and the compilation command, *compile.sh* under *Attachments*. Edit the solution template *citymapping.cpp* directly when implementing your solution, and run *compile.sh* to compile the executable. A small and large testcase for each subtask are also provided for your reference. The grader input format is provided below.

Grader Input Format

- one line with three integers: N, the number of road junctions, Q, the maximum number of queries allowed, and S, the subtask number.
- $N - 1$ lines, with the i-th line containing three integers: A_i, B_i, W_i.