# Lab: Problem Solving

This document defines the **in-class exercise** assignments for the ["Algorithms" course @ Software University](https://softuni.bg/opencourses/algorithms).

## Elections

You are given the results from the elections. There are **N** **parties** that have enough votes and are given seats in the parliament. You are given the seats for each one of the parties. For the parties to have **majority** in the parliament they **need at least K seats** (that means **K** or more seats). Parties can combine with each other in order to have **K** or more seats together.

Write a program to find the number of all possible combinations of parties with sum of seats **K** or more.

### Input

* The input data should be read from the console.
* On the first input line, there will be the number **K**.
* On the second input line, there will be the number **N.**
* On each of the next **N** lines there will be the number of the seats for each of the **N** parties.
* The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

* The output data should be printed on the console.
* On the only output line write the number of all possible combinations of parties with sum of seats **K** or more.

### Constraints

* **N** will be an integer between 1 and 100, inclusive.
* The number of seats for each party will be an integer between 1 and 1000, inclusive.
* **K** will be an integer between 1 and 100 000, inclusive.
* Allowed working time for your program: 0.30 seconds. Allowed memory: 32 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 10  3  10  4  2 | 4 | If we name the parties A(10), B(4) and C(2), then the number of all possible combinations of parties with sum of seats 10 or more is exactly **4**:  A (10), AB (14), ABC (16), AC (12) |
| 121  8  84  39  38  23  19  15  11  11 | 128 | If we name the parties A(84), B(39), C(38), D(23), E(19), F(15), G(11) and H(11), then the number of all possible combinations of parties with sum of seats 121 or more is exactly **128**:  AB, ABC, ABCD, ABCDE, ABCDEF, ABCDEFG, ABCDEFGH, ABCDEFH, ABCDEG, ABCDEGH, ABCDEH, ABCDF, ABCDFG, ABCDFGH, ABCDFH, ABCDG, ABCDGH, ABCDH, ABCE, ABCEF, ABCEFG, ABCEFGH, ABCEFH, ABCEG, ABCEGH, ABCEH, ABCF,  …  BCDFG, BCDFGH, BCDFH, BCDGH, BCEFG, BCEFGH, BCEFH |

## Sum To 13

You are given three number **a, b** and **c** as an input. Your task is to tell if the **three numbers can be summed to** **13**, by only **changing their signs**.

### Input

* On the first line, you receive the numbers **a, b** and **c**, separated by spaces.

### Output

* Print "**Yes**" if the numbers can be summed to 13, or "**No**" if it is impossible.

### Constraints

* **a**, **b**, **c** will be whole numbers in the range **[-1 000 000…1 000 000].**
* Allowed time: **100 ms**. Allowed Memory: **16 MB**.

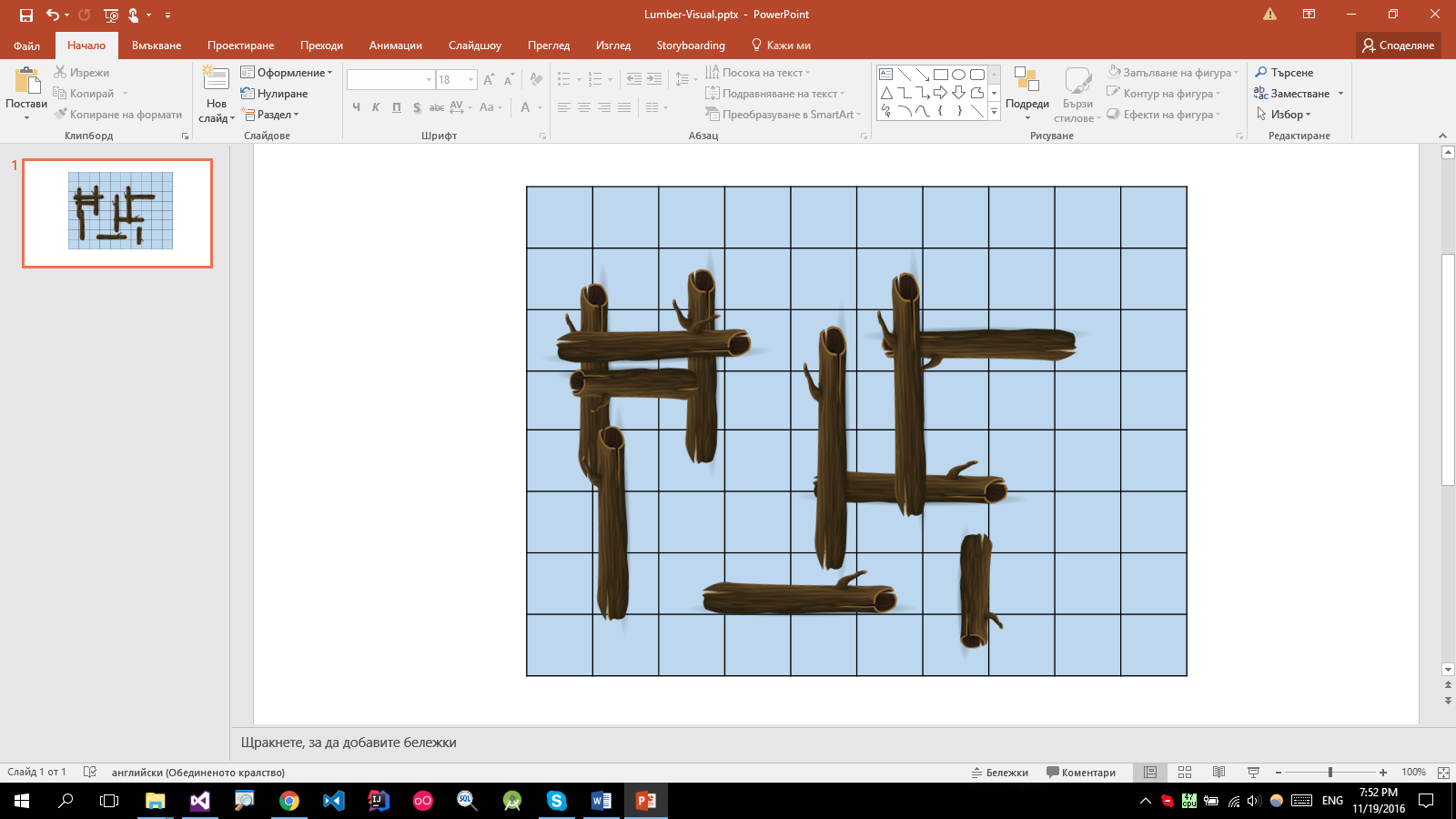
### Examples

|  |  |  |
| --- | --- | --- |
| ****Input**** | ****Output**** | ****Comment**** |
| **-10 2 -1** | **Yes** | We switch the signs of -10 and -1 so that we get 10 + 2 + 1 = 13 and thus print "Yes" |

|  |  |  |
| --- | --- | --- |
| ****Input**** | ****Output**** | ****Comment**** |
| **-1 10 -1** | **No** | Regardless of what sign switching we do, we can't get to 13, so we write "No" |

## Lumber

The Iskar river has **lumber logs** floating in its waters (see image below). Habibi is a beaver who wants to know if he can move from log **X** to log **Y**.



A log is a **rectangle**. It is defined by two corners: **top-left A**(**Ax**; **Ay**) and **bottom-right B**(**Bx**; **By**). Habibi can travel between two logs if they touch each other (their coordinates intersect).

Write a program that tells Habibi if he can travel between two arbitrary logs.

### Input

* On the first input line you will be given the number of logs **N** and the number of queries **M** as 2 space-separated integers.
* On the next **N** lines you will be given the coordinates of each log in the format "**Ax Ay Bx By**".
* On the next **M** lines you will be given queries in the format "**X Y**" where **X** and **Y** correspond to logs in the **order** they were given in the input (starting from **1**).

### Output

* For each query print "**YES**" if the two logs are connected. Otherwise, print "**NO**".

### Constraints

* The number of logs **N** will be an integer in the range **[2..1000]**.
* The number of queries **M** will be an integer in the range **[1..10000]**.
* All log **coordinates** will be valid integer numbers in the range **[-100..100]**.
* Time limit: **100 ms**. Allowed memory: **16 MB**.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Visual** |
| 4 3  -10 30 60 10  -50 20 -30 -20  -35 60 -20 15  -40 -10 50 -30  4 2  3 4  4 1 | YES  YES  NO |  |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Visual** |
| 3 3  0 50 30 40  30 50 60 40  40 40 60 1  1 2  2 3  3 1 | YES  YES  YES |  |