# Task 1

## Variation A

Goal of task 1 variation A was to get information about how many other students does each student comes to contact with for purpose of sharing/spreading information. First step to solve this issue was to read list of towns which were used to create empty adjacency matrix (matrix full of 0’s, and main diagonal was filled with 1’s). Next step was to read input line by line. Each line containing two values divided by “ – “ character. After separating two values they were used as position pointers in matrix, to set appropriate 2 values in matrix as 1’s. 2 values were set because this matrix represents unoriented graph. After reading all lines, we have our adjacency matrix ready and it now represents possible information spread, where 0’s represent fact that information will not be spread to that person and 1’s representing fact, that it will be spread to that person. When we count 1’s in each row or column and minus 1 (itself), we get number of students that information will spread to from that one person.

## Variation B

Goal of task 1 variation B was to get best possible combination of 3 people to spread information. For this purpose, we used adjacency matrix from task 1 variation A. And created combinations of all students in 3 groups. First found best combination is printed to stdout.

# Task 2

Goal of task was to merge 2 corporations and its delivery routes. We can image that each corporation represents matrix and it is values are its delivery routes. All that needs to be done for these two matrixes to be merged into one. So first we read delivery stops and create with them new matrix, repeated delivery stops are ignored. Now we have merged matrix. All that’s left is to fill it with values, one by one. While filling matrix with values, if delivery route was not in matrix then it is added to list of routes which will be used, and if it is already in matrix, then it is added to list of routes which will be cancelled. At the end of program these both delivery routes are printed to stdout.

# Task 3