Problem 1 (2 points)

In the management of the social network ThinksThat (www.thinksthat.com, where you can hear what your contacts think) they want to determine the degree of connection between their users. The degree of connection is the number of user groups connected to each other divided by the total number of users. If this value is 1, it means that the users are isolated and do not have common contacts, as the lower is the value the degree of connection of the users is higher.

Among the thousands of ThinksThat users, a random population of one hundred identifiers is chosen to obtain their degree of connection. Each of these hundred identifiers represents a user, whose contacts are also stored in a list. For example, you could have the following case of six users (in italics) and their contacts (separated by commas):

Antonio: Bea, Carlos, Emma	*Carlos:* Antonio, Emma	*Emma:* Bea
Bea: Emma, Carlos, Antonio	*David:* Fernando	*Fernando:* NONE

Although Emma only has Bea as contact, through her she is related to Carlos and Antonio. On the other hand, Fernando and David also have a connection between them (David has Fernando among his contacts, although the reciprocal is not met). However, the two groups have no common links.

In this example the degree of connection would be 2 groups divided by 6 users, approximately the value 0'33.

Design a **voracious** algorithm that, having as data the chosen users (each of them with their identifier and the list of their contacts), obtain the degree of connection between them.

Delivery requirements:

- Deliver a report in which the solution is explained, justifying why the implementation has been chosen.
- Source code.
- Input file ‘example_voracious.txt’
- Executable that can be tested by inserting the example file, “example_voracious.txt”, and returns an output file with the number of groups, the members of these groups and the degree of connection. The executable must work in any Windows environment. The example file must be read from the same folder where the executable is located.

Problem 2 (1 point).

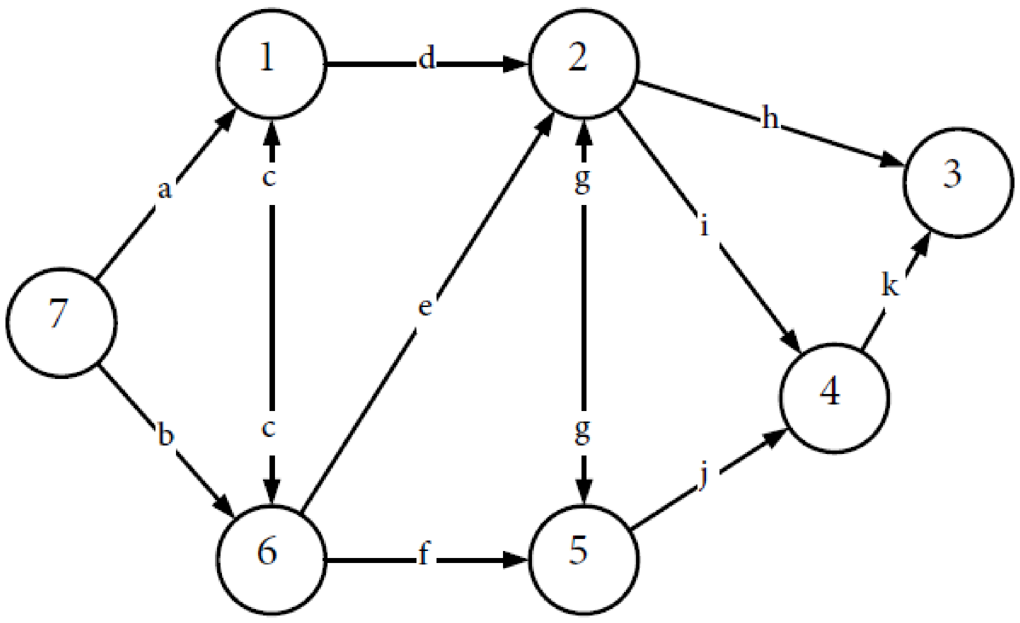
Calculate the average grade of a student in a course consisting of n subjects using the **divide and conquer** technique, with n being a power of 2. Modify the algorithm to allow values of n that are not a power of 2. Calculate efficiency and determine if it improves the that is obtained with respect to a simple iterative version.

Delivery requirements:

- Deliver a report in which the solution is explained, justifying why the implementation has been chosen.
- Source code.
- Input file 'example_d&c.txt'
- Executable that can be tested by inserting the example file, "example_d&c.txt", and returns an output file with the average value obtained. The average value obtained in all recursive stages of the divide and conquer process must be shown. The executable must work in any Windows environment. The example file must be read from the same folder where the executable is located.

Problem 3 (0.75 points).

Given the following graph:



It is requested:

a) Generate 11 random numbers between 1 and 10.
b) Substitute the letters of the graph axes for the generated random numbers.
c) Apply the Floyd algorithm and obtain the minimum distance matrix of the graph. Show all intermediate iterations of the algorithm.

Problem 4 (2 points).

Bruce Springsteen is going to finish his world tour this year giving the last concert in Abecelandia, a city famous for its beautiful squares and that you may know, so a truck loaded with instruments, sound devices, flares, stands and nuts has entered in the city and heads to the stadium to set up the stage and fireworks. Suddenly, one of the front wheels explodes and partially renders the truck's steering system useless, preventing it from making left turns. Due to its size and weight, the truck cannot go in reverse (so it cannot go backwards) or change direction on the same street (that is, it cannot make a 180º turn without changing streets), so now the truck can only drive in a straight line and make 90º turns to the right. Fortunately, the driver has a map of Abecelandia like the following, where point 1 is the current position of the truck, point 2 is the stadium to go to, and each position is marked as accessible (if empty) or non-accessible (if it is in black). It can be assumed that initially the truck is oriented to the left, and that the map is FxC size. Therefore, at each position on the map, the truck can continue moving (assuming that the next space is free) or turn 90º to the right (once again, assuming that position is accessible). Since it is not known if the rest of the wheels will hold until reaching the stadium, we want to minimize the number of turns so as not to overload the tires that are still working. Implement an algorithm with **Backtracking** methodology, efficient as far as possible, taking as data the sizes in rows F and columns C of the map, the MFxC map and the coordinates of points 1 and 2, which finds the path that leads from 1 to 2 with the fewest turns to the right.

IMPORTANT: It is essential to comply with the delivery requirements and the input formats.

Input format: a file "example_backtracking.txt", with the following format:

- Line 1: Number of rows (F).
- Line 2: Number of columns (C).
- Line 3: Description of the first row (C characters separated by tabs).
- Line 4: Description of the second row (C characters separated by tabs).
- …
- Line 2 + F: Description of the last row (C characters separated by tabs).

Description of possible characters:

- '1': Start on the map.
- '2': End on the map.
- '0': Free space on the map, through which you can circulate.
- 'x': occupied square on the map, which cannot be circulated.

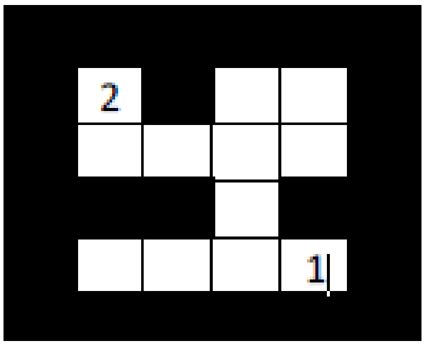
Delivery requirements:

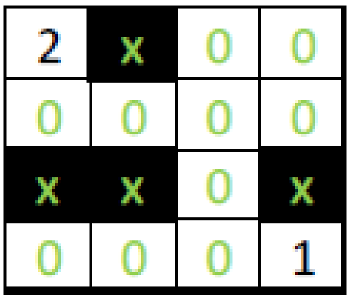
- Deliver a report in which the solution is explained, justifying why the implementation has been chosen.
- Source code.
- Executable that can be tested by inserting any input file, "example_backtracking.txt", and returns an output file with all the solutions

found, as well as the optimal one. The specific route for each solution must be specified (for example, indicating whether in each square through the truck passes it continues straight or turns to the right). The executable must work in any Windows environment. The example file must be read from the same folder where the executable is located.

Examples:

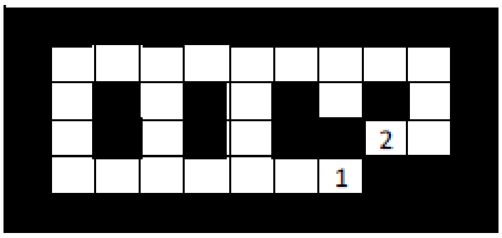
"Example1.txt":

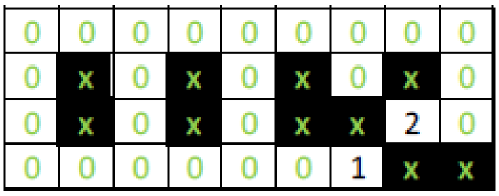




2	x	0	0
0	0	0	0
x	x	0	x
0	0	0	1

“Example2.txt”





0	0	0	0	0	0	0	0	0
0	x	0	x	0	x	0	x	0
0	x	0	x	0	x	x	2	0
0	0	0	0	0	0	1	x	x

Problem 5 (0.75 points).

The following table corresponds to an allocation problem:

	1	**2**	**3**	**4**
a	a_{11}	a_{12}	a_{13}	a_{14}
b	a_{21}	a_{22}	a_{23}	a_{24}
c	a_{31}	a_{32}	a_{33}	a_{34}
d	a_{41}	a_{42}	a_{43}	a_{44}

It is requested:

a) Generate 16 random numbers between 5 and 35.
b) Substitute the a_{ij} values for the generated numbers.
c) Solve the corresponding allocation problem using branch and bound.