ALGORITHMICS - Bachelor of Software Engineering, 2024

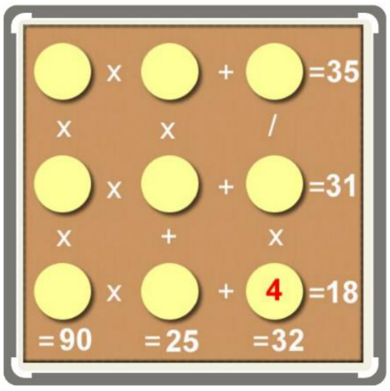
LAB GUIDE. SESSION 7

GOALS:

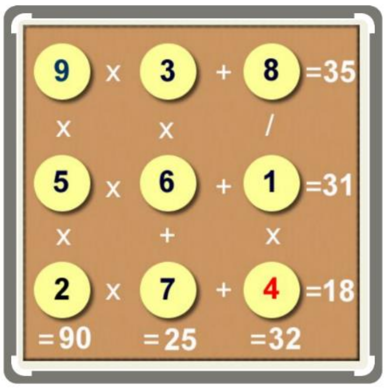
- **Branch and bound algorithms**

1. The numerical square

The numerical square is a mental agility game that is based on a board with certain mathematical operations (addition, subtraction, multiplication and division) both horizontally and vertically. The objective is to fill in the missing numbers so that the results are true. Also, as help, some numbers are already entered on the board. Below is an example of a 3x3 size starter board:



After some thinking, we could obtain a solution like the following, in which all the results become true after adding the corresponding numbers:



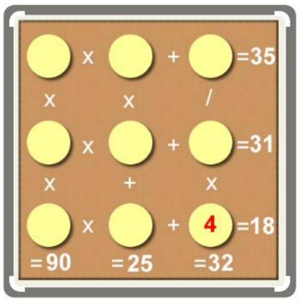
It must be considered that in the best-known version of the game you only have to add all the numbers from 0 to 9 without repeating any of them, which greatly simplifies the solution to the problem.

In **our version** of the game there will be three important rules:

1) The numbers to be entered will be from 0 to 9, and any of them can be repeated without any limit of times.
2) It is not necessary to use all numbers from 0 to 9.
3) The order of precedence of mathematical operations will be the order of appearance of the operators. That is, for example 2+3*5 will be equal to 25, not 17 as we might think.

The program will try to solve the board that will be passed to it in a text file. As an example, you can see the text file that represents the board shown above:

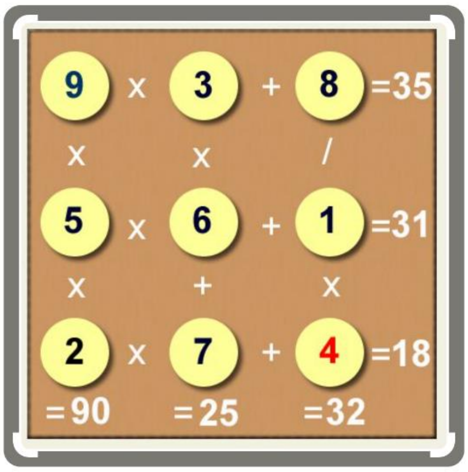
```
3
? * ? + ? = 35
* * /
? * ? + ? = 31
* + *
? * ? + 4 = 18
= = =
90 25 32
```



Representing the following:
- First line: **size of the board** (3x3 in the example).
- Second and subsequent lines: **information about operations.**
    - Note that the holes to be completed with numbers are represented with a **?**

The program must show each of the results in the console, clearly displaying the results obtained and the operations performed, as shown below for the same example:

```
SOLUTION FOUND
         9     *     3     +     8     =     35

         *           *           /

         5     *     6     +     1     =     31

         *           +           *

         2     *     7     +     4     =     18

         =           =           =

        90          25          32
```



YOU ARE REQUESTED TO:

Design and implement an algorithm using the Branch and Bound technique to solve this problem optimally. The objective is to find a heuristic that, as far as possible, improves the times obtained with backtracking.

Implement said algorithm in Java (NumericSquareBaB.java) in such a way that it calculates a solution for a given board entry in the most efficient way possible.

Explain what heuristic has been used to quantify the quality of each node (that is, to quantify how close to a solution that node might be).

Fill in the following table:

TABLE 1

(times in milliseconds and WITH OPTIMIZATION):

We can type "OoT" for times over four minutes and "LoR" for times less than 50 milliseconds.

Test case	*Time for first solution (backtracking)*	*Number of developed nodes (backtracking)*	*Time for first solution (branch and bound)*	*Number of developed nodes (branch and bound)*
Test00				
Test01				
Test02				
Test03				
Test04				
Test05				
Test06				
Test07				

What algorithm has worked in a better way? Why do you think this has happened?

2. Work to be done

- An `algstudent.s7` **package** in your course project. The content of the package should be the Java files used and created during this session.
- A `session7.pdf` **document** using the course template (the document should be included in the same package as the code files). You should create one activity each time you find a "YOU ARE REQUESTED TO" instruction.

Deadline: The deadline is one day before the next lab session of your group.