***Logical reasoning and programming*, task III**
(December 12, 2022)

Problem

Choose one of the two logic puzzles described below and implement a search-based solver for it in Prolog.

- You can choose between an easier version, where you implement a solver for one particular puzzle described below, or you can go for a harder version, implementing a solver for the puzzle's generalized version (described in the *Generalization* section).

  The easier version may yield at most 10 points. The harder one may yield the full 15 points obtainable from this task.

- Two search strategies are expected, chosen from three options:

  – depth-first search,
  – iterative-deepening depth-first search
  – breadth-first search

  Each strategy will give you 50% of the maximum achievable points.

- You are expected to use only the techniques covered by this course. In particular, implementations using a constraint-satisfaction library (CSP) or definite clause grammar (DCG) will not be accepted.

1 Escape from Zurg

Buzz, Woody, Rex, and Hamm need to escape from Zurg. They merely have to cross one last bridge before they are free. However, the bridge is fragile and can hold at most two of them at the same time. Moreover, to cross the bridge, a flashlight is needed to avoid traps and broken parts. The problem is that our friends have only one flashlight with one battery that lasts for only 60 minutes. The toys need different times to cross the bridge (in either direction):

Toy	Time
Buzz	5
Woody	10
Rex	20
Hamm	25

Since there can be only two toys on the bridge at the same time, they cannot cross the bridge all at once. Since they need the flashlight to cross the bridge, whenever two have crossed the bridge, somebody has to go back and bring the flashlight to those toys on the other side that still have to cross the bridge. The problem now is: In which order can the four toys cross the bridge in time (that is, within 60 minutes) to be saved from Zurg?

One solution is:

```
[left_to_right(buzz,woody), right_to_left(buzz),
left_to_right(hamm,rex), right_to_left(woody),
left_to_right(buzz,woody)].
```

Generalization: Implement a search-based solver for a generalized version of the "Escape from Zurg" problem using Prolog. Namely, implement a predicate such that

- 1^{st} argument is a time-limit, which is a positive number T,
- 2^{nd} argument is list of toys and their times to cross the bridge and
- 3^{rd} argument is a plan how the toys may escape within given time limit.

```
?- escape_zurg(60,[[buzz,5],[woody,10],[rex,20],[hamm,25]], Sol).
Sol = [left_to_right(buzz,woody), right_to_left(buzz),
left_to_right(hamm,rex), right_to_left(woody),
left_to_right(buzz,woody)] ;
...
```

2 Verbal Arithmetic

Find an assignment of numbers to letters D, E, M, N, O, R, S and Y so that the following equation holds:

$$\begin{array}{cccccc} & S & E & N & D \\ + & M & O & R & E \\ \hline M & O & N & E & Y \end{array}$$

Avoid trivial solutions, where all letters are assigned the value 0.

One solution is:

```
D = 7, E = 5, M = 1, N = 6, O = 0, R = 8, S = 9, Y = 2.
```

$$\begin{array}{cccccc} & 9 & 5 & 6 & 7 \\ + & 1 & 0 & 8 & 5 \\ \hline 1 & 0 & 6 & 5 & 2 \end{array}$$

Generalization: Implement a search-based solver for the "Verbal Arithmetic" problem using Prolog. Namely, implement a predicate such that the input consists of any 3 words in the English alphabet. The first 2 words have equal length, and the third word is longer by 1 character. We seek a replacement of each letter by a number from 0 to 9 so that the first two words are summed to the third one, avoiding trivial solutions.

```
?- sum([S,E,N,D], [M,O,R,E], [M,O,N,E,Y])
D = 7, E = 5, M = 1, N = 6, O = 0, R = 8, S = 9, Y = 2 ;
...
```

Evaluation

Your solutions will be evaluated by hand. You may obtain up to 15 points.

You should upload an archive containing your solutions to BRUTE before deadline. Keep your code in a single file that can be loaded into Prolog. Your code must satisfy a few requirements. It

1. must have clear instructions as how to run it in SWI Prolog and how to read the results (0% of the grade),
2. should implement the specified behaviour without reservations; it should find a solution if and only if the solution exists (70%),
3. should be legible, well documented and easy to understand (20%),
4. should show no compiler warnings when loaded (10%).

The requirements must be satisfied in order of appearance! E.g., if you satisfy $1, 3, 4$ without satisfying 2, you will not receive any points for 3 and 4.