Logical reasoning and programming, **task III**

(December 18, 2023)

Problem

Consider the logic puzzle below. Implement a search-based solver for it in Prolog.

You are expected to use only the techniques covered by this course. Submited solution should be your own creation.

Puzzle: Escape from Zurg

A group of toys need to escape from the Emperor Zurg. They merely have to cross one last bridge before they are free. However, the bridge is fragile and it can only hold two toys at the same time. Moreover, to cross the bridge, a flashlight is needed to avoid traps and broken parts. The problem is that the toys have only one flashlight with battery that lasts only a limited amount of time. Additionally, each toy moves at different speed and if two toys are crossing the bridge together, they move at the speed of the slower one.

Since there can only be 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 of them 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 toys cross the bridge in time (i.e., within the lifespan of the flashlight battery) to be saved from Zurg?

Puzzle: Example

Buzz, Woody, Rex, and Hamm are trying to escape from Zurg. Their flashlight will only last 60 minutes. Toys' movement speeds (in minutes) are as follows:

Toy	Speed
Buzz	5
Woody	10
Rex	20
Hamm	25

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)].
```

Implementation

Write a predicate `escape_zurg/3` in Prolog that finds all possible escape solutions. The solutions should be returned in a lazy fashion, one at a time. The predicate's arguments should be as follows:

- 1st argument is a time-limit (battery lifespan), which is a non-negative number T,
- 2nd argument is list of toys and their times to cross the bridge,
- 3rd 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)] ;
...
```

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 points),
2. should implement the specified behaviour without reservations; it should find a solution if and only if the solution exists and it should terminate when there are no (more) solutions (10 points),
3. should be reasonably efficient, e.g., do not return the same solutions written differently (2 points)
4. should be legible, well-documented and easy to understand (2 points),
5. should show no compiler warnings when loaded (1 point).

In order to receive points for items $3, 4$ or 5, you must first satisfy items 1 and 2 (and obtain more than zero points from item 2).