CvP
Programming Assignment 2
Prolog

For this exercise, we use the SWI-Prolog (http://www.swi-prolog.org/) implementation of the Prolog language. The Prolog interpreter has been compiled for Linux/x86 systems, and isavailable on the LIACS network at `/home/csalp/bin.linux/prolog/pl`. Versions for other systems are available for download at the aforementioned web-page.

You can run the interpreter as follows:

```
/home/csalp/bin.linux/prolog/pl
```

or run the interpreter for a given file:

```
/home/csalp/bin.linux/prolog/pl -f myprog.pl
```

where `myprog.pl` is your Prolog program.

Note: when using the first version, type `consult('myprog.pl').` at the beginning and every time you change the program.

Once you have started the interpreter, you can test your program or edit it. You can test Prolog by typing a goal:

```
?-conc([1,2,3,4,5],[6,7],X).
```

Assuming `conc` is defined as expected, the interpreter will produce:

```
X=[1,2,3,4,5,6,7]
```

Now there are two possibilities: you can enter a `;` to let the interpreter explore alternative values for X. If none are found, it will produce `No`. By just pressing the enter key the interpreter will stop searching for alternative values for X. The interpreter will produce a `Yes` and you can enter a new command. If you want to edit your program, enter: `?-edit.` This option will only work if you run Prolog using the `-f` option. Otherwise, enter: `?-edit('myprog.pl').`

Some general remarks:

- Use capital letters when using variables: use `Name` instead of `name`;
- When using functions, do not put a space between the function name and the first bracket: `voegsamen(\ldots)` instead of `voegsamen (\ldots);`
- Put a `.` at the end of every command, or they will not be executed;
- To use the implication symbol (⟵) in SWI-Prolog, you have to use `:-`, i.e., for: bird(X) ⟵ lays eggs(X) ∧ has wings(X) we write: `bird(X):-lays_eggs(X),has_wings(X).`
- You can exit SWI-Prolog by entering CTRL-C followed by an `e`.

Exercises

Binary Search Trees

Consider binary trees whose nodes are labeled with natural numbers, and use the term `void` to denote the empty tree, and the term `tree(x, left, right)` to denote the tree with root `x`, left subtree `left` and right subtree `right`.

For example, the term

```
tree(1,tree(2,void,void), tree(3,void,void))
```

represents the tree with root 1 and children 2 and 3. We call a binary tree `tree(x,left,right)` ***nice*** if both of following statements hold:

- If `left` is not empty, then `x` is greater than all the elements in `left`;
- If `right` is not empty, then `x` is less than all the elements in `right`.

A binary tree is called a ***search tree*** if each of its sub-trees is nice.

Write a program which tests whether a ground term is a search tree.

Hint: Use the following predicate `is_search_tree(T)` in the definition of search tree:

```
is_search_tree(void).is_search_tree(T):-
is_search_tree(T,Min,Max).
```

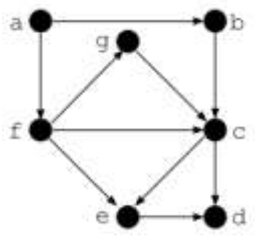
where Min and `Max` are the minimum and maximum element of the tree `T`. Then implement the predicate `is_search_tree(T,Min,Max).`

List length

Write a program in Prolog to find a length of a given list. For example, goal `length([a, b, c, d, e])` should print `5`.

Graph path search

Write a Prolog program which describes a directed graph with the following structure:



Define the path relation `path(Node1,Node2):-`… on this graph.

Submission Instructions

Your programs should be submitted together with a written report in which you explain your programs to "Sander van Rijn" svr003@gmail.com.

The report can only be submitted as plain text or PDF; other formats than `.txt` and `.pdf` will not be accepted. Remember to include your full name and student number in the body of the email.