German University in Cairo
Media Engineering and Technology
Dr. Nourhan Ehab

CSEN 403: Concepts of Programming Languages, Spring Term 2024
Practice Assignment 3

Exercise 3-1 Arithmetic and Recursion

Write the following Prolog predicates:

a) `maximum(A,B,C)` succeeds if `C` is the maximum of `A` and `B`.

b) `seq(First, Last, N)` succeeds if `N` is an integer between `First` and `Last`. The predicate should list all integers from `First` through `Last`, inclusively. Assume that `First` and `Last` are integers.

```
?- seq(1,3,N).
N = 1 ?;
N = 2 ?;
N = 3 ?;
no
```

c) `mult(X,Y,Z)` succeeds if `Z` is the result of the multiplication of `X` by `Y`. You are not allowed to use the Prolog's built-in multiplication operation.

Hint: The multiplication function can be defined as follows:

$$mult(x,y) = \begin{cases} 0 & : \quad \text{if } y = 0 \\ x + mult(x, y-1) & : \quad \text{if } y > 0 \end{cases}$$

d) `expo(X,Y,Z)` succeeds if `Z` is the exponent of `X` and `Y`. You are not allowed to use the Prolog's built-in multiplication or exponention operations.

Hint: The exponent function can be defined as follows:

$$expo(x,y) = \begin{cases} 1 & : \quad \text{if } y = 0 \\ mult(x, exp(x, y-1)) & : \quad \text{if } y > 0 \end{cases}$$

Exercise 3-2 Arithmetic and Recursion

Implement the predicate `ackermann(X,Y,Z).` which succeeds if `Z` is the result of applying Ackermann's function on `X` and `Y` according to the following mathematical definition:

```
f(0,y)=y+1
f(x,0)=f(x-1,1) if x>0
f(x,y)=f(x-1,f(x,y-1))if x,y>0
```

Exercise 3-3 Arithmetic

a) Write twelve Prolog facts, one for each month of the year. Each fact contains three pieces of information:

[0]The exercises are due to Prof. Dr. Slim Abdennadher and Dr. Nada Sharaf.

- the number of the month (1-12)
- the name of the month (january, february, ..., december)
- the number of days in the month

b) Write a Prolog rule called `days_left(Month, Day, DaysLeft)` that uses the facts in the part a) to calculate the number of days left in a month.

Exercise 3-4

Consider a knowledge base in which the course is defined using the following predicate:

```
course(Name,Timing,Lecturer,Location).
```

where:

- The name of the course is defined as a constant.
- The course timing is defined by the structure `timing(Day,Slot)`
- The course lecturer is defined by the structure `lecturer(First_name,Last_name)`
- The course location is defined by the structure `room(Building,Room_number)`

For example, we may have:

```
course(csen403,timing(wednesday,1),lecturer(slim,abdennadher),room(c,h14)).
course(csen401,timing(monday,1),lecturer(slim,abdennadher),room(b,h3)).
```

Based on this representation, define the following predicates:

- `schedule(Room_number,X)` where `X` is the name of a course scheduled in `Room_number`.
- `busy(Room_number,Day,Slot)` which checks whether the room `Room_number` is busy on the day `Day` at slot `Slot`.

Exercise 3-5 Binary Trees

Write a prolog predicate depth(T,D) that holds if D is the depth of tree T, i.e. the length of the longest branch).
The tree is of the form node, left subtree, and right subtree. For example:

```
bt(5, bt(6, nil, nil), bt(8,bt(9,nil, nil),nil))
```

An empty binary tree is considered to be of depth zero.

Exercise 3-6 Successor Notation

Integers and floating point numbers are built into most programming languages, including Prolog. However, suppose that numbers and arithmetic operations were not available. It is still possible to define numbers in Prolog and to write programs to implement simple arithmetic operations. In the following question, you MUST NOT use any of Prolog's built-in arithmetic predicates such as `is, <, +, *`, etc. (except in part (f)).

Integer numbers can be represented based on the constant `0` and a successor function. A positive integer is defined as zero or the successor of another number. Thus, zero can be represented by `0` and any number, such as two, can be represented by `s(s(0))`, where two is the successor of the successor of `0`. Thus `s(X)` can be thought of as `X+1`.

a) Write a Prolog Predicate, `sum(X, Y, Z)` which is true if `Z` represents the result of adding `X` to `Y`.

```
?- sum(s(s(0)), s(0), Z).
Z=s(s(s(0)))

?- sum(s(s(s(0))), s(s(0)), Z).
Z=s(s(s(s(s(0)))))
```

b) Write a Prolog predicate, `subtract(X, Y, Z)`, which is true if `Z` represents the difference between `X` and `Y`, assuming `X` is greater than or equal to `Y`. If `X` is less than `Y`, then this predicate is undefined.

```
?- subtract(s(s(s(s(0)))), s(s(0)), Z).
Z = s(s(0))
```

Hint: The base case is when `Y=0`.

c) Write a Prolog predicate, `mult(X, Y, Z)`, which is true if `Z` represents the multiplication of `X` and `Y`. **Do not use *!**

Exercise 3-7 Evaluate Expressions

Implement the predicate `evaluate(Expr,Result)` that evaluates arithmetic expressions. The expression to be evaluated will be described as a structure as follows:

- **plus(X,Y):** The structure representing the addition of the two expressions X and Y.
- **minus(X,Y):** The structure representing the subtraction of expression Y from expression X.
- **times(X,Y):** The structure representing the multiplication of the two expressions X and Y.
- **divide(X,Y):** The structure representing the division of the expression X by the expression Y.

Note that the arguments of any of these structures can be a mathematical expression or a number.

Example:

```
?- evaluate(divide(plus(minus(6,4),times(2,3)),times(1,2)),Z).

Z = 4

Yes
```

Hint: You may find it useful to use the predefined prolog predicate `number(X)` which evaluates to true if the parameter X is a number.