Lab Exercises - Structure Inspection

CMPT333N

Problem 1

Write an `add_matrices/3` predicate which can add matrices of arbitrary dimensions. Matrices will be represented using the functor `mat/3`, and are implemented by allowing the arguments of `mat/3` to be themselves matrices, and not only numbers. For example, the structure

```
mat(mat(1, 2, 3), mat(4, 5, 6), mat(7, 8, 9))
```

would represent the matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Problem 2

Write predicate `replace_var_args(Term, Values, NewTerm)` which is true if it replaces the terms of Term which are variables by values from the list ≪Values≫. That is, the 1st variable of `Term` is replaced by the 1st term of the list `Values`, the 2nd variable of `Term` is replaced by the 2nd term of the list Values and so on until the list Values becomes empty. Finally, it returns the new term `NewTerm`. Use the built-in predicates like `functor/3, =../2, var/1, nanvar/1`, etc. Examples of goals,

```
?- replace_var_args(father(yannis,X), [anna], NewTerm).
NewTerm = father(yannis,anna).

?- replace_var_args(f(a,X,g(b), Y,h(a,Z)), [b,c], NewTerm).
NewTerm = f(a,b,g(b),c,h(a,Z)).

?- replace_var_args(f(a,X,f(b),Y,h(a,Z)), [b,c,d], NewTerm).
NewTerm = f(a,b,g(b),c,h(a,d)).

?- replace_var_args(t(X, Y, Z,a,f(Z,b)), [1,2,3], NewTerm).
NewTerm = t(1,2,3,a,f(Z,b)).
```

Problem 3

Write a Prolog program to print all Pythagorean triples (x, y, z) such that $1 \leq x$, $y \leq z \leq 100$. ((x, y, z) is a Pythagorean triple if x * x + y * y = z * z.)

Problem 4

Rewrite the program below so that it counts down. (*Hint*: Use an accumulator.)

```
Term =.. [F|Args]:-
    functor(Term,F,N), args(0,N,Term,Args).
args(I,N,Term,[Arg|Args]):-
    I < N, I1 is 1+1, arg(I1,Term,Arg), args(I1,N,Term,Args).
args(N,N,Term,[]).
```

Problem 5

Define `functor` and `arg` in terms of `univ`. How can the programs be used?

Problem 6

Rewrite the program for `substitute` so that it uses `univ`.

```
substitute(Old,New, Old,New).
substitute(Old,New,Term,Term) :-
    constant(Term), Term \= Old.
substitute(Old,New,Term,Termi) :-
    compound(Term),
    functor(Term,F,N),
    functor(Terml,F,N),
    substitute(N,Old,New,Term,Terml).

substitute(N,Old,New,Term,Terml) :-
    N > 0,
    arg(N,Term,Arg),
    substitute(Old,New,Arg,Argl),
    arg(N,Terml,Argl),
    Nl is N-1,
    substitute(N1,Old,New,Term,Term1).
substitute(0,Old,New,Term,Term1).
```