Lab 10: Programming in Prolog

Baheem Ferrell

Exercise 8 - Recursion Exercise

Which of the following programs uses recursion.

```
a(X):-
b(X,Y),
a(X).
```

YES

```
go_home(no_12).
go_home(X):-
        get_next_house(X,Y),
        home(Y).
```

NO

```
foo(X):- bar(X).
```

NO

```
lonely(X):- no_friends(X).
no_friends(X):- totally_barmy(X).
```

NO

```
has_flu(rebecca).
has_flu(john).
has_flu(X):- kisses(X,Y),has_flu(Y).
kisses(janet,john).
```

YES

```
search(end).
search(X):- path(X,Y), search(Y).
```

YES

Recursion Exercise 2

1. Given facts such as

Bob is taller than Mike.

Mike is taller than Jim

Jim is taller than George

Write a recursive program that will determine that Bob's height is greater than George's.

```
taller_than_rule(X,Y) :-
    taller_than(X,Y).
taller_than_rule(X,Y) :-
    taller_than(X, Z),
    taller_than_rule(Z, Y).
```

taller_than_rule(bob,Who). Who = mike ; Who = jim ; Who = george ; false.

Recursion Exercises 3

town1----->-----town2---->----town3---->----town4--->----town5---->---town6

A one way road links 6 towns. Write a program that can work out if you can travel on that road. For example. Here are two sample program behaviours.

?- can_get(town2,town5).

yes

?- can_get(town3,town1).

No

connected(town1,town2).

connected(town2,town3).

connected(town3,town4).

connected(town4,town5).

connected(town5,town6).

can_get(X,Y) :- connected(X,Y).

can_get(X,Y) :- connected(X,Z), can_get(Z,Y).

Exercise 9 - Lists Exercise

Here are some problems for which unification sometimes succeeds and sometimes fails. Decide which is the case and, if you think the unification succeeds, write down the substitutions made.

1. [a,d,z,c] and [H|T]

YES
2. [apple,pear,grape] and [A,pear|Rest]
YES
3. [a|Rest] and [a,b,c]
YES
4. [a,[]] and [A,B|Rest]
YES
5. [One] and [two|[]]
YES
6. [one] and [Two]
YES
7. [a,b,X] and [a,b,c,d]
NO

List Construction Exercises 1
Write a program to delete all references of a particular item from a list. It should have three arguments. The list you wish to use, the item to delete, and the resulting list. Here are some example of its behavior

?- delete_all([a,b,a,c,a,d],a,Result).

Result = [b,c,d]

?- delete_all([a,b,a,c,a,d],b,Result).

Result = [a,a,c,a,d]

?- delete_all([a,b,a,c,a,d],prolog,Result).

Result = [a,b,a,c,a,d]

```
delete_all([],_,[]).
delete_all([X|T],Y,[X|Result]):- dif(X,Y), delete_all(T,Y,Result).
delete_all([Y|Tail],Y,Result):- delete_all(Tail,Y,Result).
```

PROLOG CODE

```
delete_all([],_,[]).
delete_all([X|T],Y,[X|Result]):- dif(X,Y), delete_all(T,Y,Result).
delete_all([Y|Tail],Y,Result):- delete_all(Tail,Y,Result)
```

List Construction Exercises 2

Write a program to replace all occurrences of one item in a list with another. It should have four arguments. The list you wish to use. The item to replace. The item to replace it with, and the resulting list. Here are some example of its behaviour

?- replace_all([a,b,a,c,a,d],a,mike,Result).

Result = [mike,b,mike,c,mike,d]

?- replace_all([a,b,a,c,a,d],b,foo,Result).

Result = [a,foo,a,c,a,d]

?- replace_all([a,b,a,c,a,d],prolog,logic,Result).

Result = [a,b,a,c,a,d]

PROLOG CODE

```
replace_all([],A,B,[]).

replace_all([H|T],A,B,[B|Result]) :- H=A, replace_all(T,A,B,Result).

replace_all([H|T],A,B,[H|Result]) :- replace_all(T,A,B,Result)
```