2nd Prologue Assignment

Justin Cesarini

Task1

There are many things I learned with this prolog assignment. I was able to further my skills in prolog and be able to create state space solving problems such as the hanoi tower which was very interesting. Overall I enjoyed this assignment!

Task3

State Space Operator Implementation

```
m12([Tower1Before,Tower2Before,Tower3],[Tower1After,Tower2After,Tower3]) :-
Tower1Before = [H|T],
Tower1After = T,
Tower2Before = L,
Tower2After = [H|L].
```

Unit Test Code

```
test__m12 :-
write('Testing: move_m12\n'),
TowersBefore = [[t,s,m,l,h],[],[]],
trace('','TowersBefore',TowersBefore),
m12(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).
```

Unit Test Demo

```
?- test__m12.
Testing: move_m12
TowersBefore = [[t,s,m,l,h],[],[]]
TowersAfter = [[s,m,l,h],[t],[]]
```

Task 4

```
m12([Tower1Before,Tower2Before,Tower3],[Tower1After,Tower2After,Tower3]) :-
Tower1Before = [H|T],
Tower1After = T,
Tower2Before = L,
Tower2After = [H|L].

m13([Tower1Before,Tower2,Tower3Before],[Tower1After,Tower2,Tower3After]) :-
Tower1Before = [H|T],
Tower1After = T,
Tower3Before = L,
Tower3After = [H|L].

m21([Tower1Before,Tower2Before,Tower3],[Tower1After,Tower2After,Tower3]) :-
Tower2Before = [H|T],
Tower2After = T,
Tower1Before = L,
Tower1After = [H|L].

m23([Tower1,Tower2Before,Tower3Before],[Tower1,Tower2After,Tower3After]) :-
Tower2Before = [H|T],
Tower2After = T,
Tower3Before = L,
Tower3After = [H|L].

m31([Tower1Before,Tower2,Tower3Before],[Tower1After,Tower2,Tower3After]) :-
Tower3Before = [H|T],
Tower3After = T,
Tower1Before = L,
Tower1After = [H|L].

m32([Tower1,Tower2Before,Tower3Before],[Tower1,Tower2After,Tower3After]) :-
Tower3Before = [H|T],
Tower3After = T,
Tower2Before = L,
Tower2After = [H|L].
```

```
test__m12 :-
write('Testing: move_m12\n'),
TowersBefore = [[t,s,m,l,h],[],[]],
trace('','TowersBefore',TowersBefore),
m12(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).

test__m13 :-
write('Testing: move_m13\n'),
TowersBefore = [[t,s,m,l,h],[],[]],
trace('','TowersBefore',TowersBefore),
m13(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).

test__m21 :-
write('Testing: move_m21\n'),
TowersBefore = [[],[t,s,m,l,h],[]],
trace('','TowersBefore',TowersBefore),
m21(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).

test__m23 :-
write('Testing: move_m23\n'),
TowersBefore = [[],[t,s,m,l,h],[]],
trace('','TowersBefore',TowersBefore),
m23(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).


test__m31 :-
write('Testing: move_m31\n'),
TowersBefore = [[],[],[t,s,m,l,h]],
trace('','TowersBefore',TowersBefore),
m31(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).

test__m32 :-
write('Testing: move_m32\n'),
TowersBefore = [[],[],[t,s,m,l,h]],
trace('','TowersBefore',TowersBefore),
m32(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).
```

Demo

```
?- test__m12
Testing: move_12
TowersBefore = [[t,s,m,l,h],[],[]]
TowersAfter = [[s,m,l,h],[t],[]]

?- test__m21
|    .
Testing: move_m21
TowersBefore = [[],[t,s,m,l,h],[]]
TowersAfter = [[t],[s,m,l,h],[]]
true.

?- test__m23.
Testing: move_m23
TowersBefore = [[],[t,s,m,l,h],[]]
TowersAfter = [[],[s,m,l,h],[t]]
true.

?- test__m31.
Testing: move_m31
TowersBefore = [[],[],[t,s,m,l,h]]
TowersAfter = [[t],[],[s,m,l,h]]
true.

?- test__m32.
Testing: move_m32
TowersBefore = [[],[],[t,s,m,l,h]]
TowersAfter = [[],[t],[s,m,l,h]]
true.
```

```
?- test__m13.
Testing: move_m13
TowersBefore = [[t,s,m,l,h],[],[]]
TowersAfter = [[s,m,l,h],[t],[]]
```

Task 5

Code

```
disc_position([P1,P2,P3]) :-
correctspot(P1),
correctspot(P2),
correctspot(P3).
correctspot([]).
correctspot([t]).
correctspot([t,s]).
correctspot([t,m]).
correctspot([t,l]).
correctspot([t,h]).
correctspot([t,s,m]).
correctspot([t,s,m,l]).
correctspot([t,s,m,h]).
correctspot([t,s,m,l,h]).
correctspot([s]).
correctspot([s,m]).
correctspot([s,l]).
correctspot([s,h]).
correctspot([s,m,l]).
correctspot([s,m,h]).
correctspot([s,m,l,h]).
correctspot([m]).
correctspot([m,l]).
correctspot([m,h]).
correctspot([m,l,h]).
correctspot([l]).
correctspot([l,h]).
correctspot([h]).


test_production :-
write('Testing: production\n'),
test_v([[l,t,s,m,h],[],[]]),
test_v([[t,s,m,l,h],[],[]]),
test_v([[],[h,t,s,m],[l]]),
test_v([[],[t,s,m,h],[l]]),
test_v([[],[h],[l,m,s,t]]),
test_v([[],[h],[t,s,m,l]]).
```

Demo

```
?-
|    test_production.
Testing: production
[[l,t,s,m,h],[],[]] is invalid.
[[t,s,m,l,h],[],[]] is valid.
[[],[h,t,s,m],[l]] is invalid.
[[],[t,s,m,h],[l]] is valid.
[[],[h],[l,m,s,t]] is invalid.
[[],[h],[t,s,m,l]] is valid.
true
```

Task 6

Code

```
transfer_discs :-
write('First display_sequence ...'), nl,
display_sequence([m31,m12,m13,m21]),
write('Second display_sequence ...'), nl,
display_sequence([m13,m12,m32,m13,m21,m23,m13]).


transfer(m12,ToText):-
ToText = 'Transfer a disk from tower 1 to tower 2'.
transfer(m13,ToText):-
ToText = 'Transfer a disk from tower 1 to tower 3'.
transfer(m21,ToText):-
ToText = 'Transfer a disk from tower 2 to tower 1'.
transfer(m23,ToText):-
ToText = 'Transfer a disk from tower 2 to tower 3'.
transfer(m31,ToText):-
ToText = 'Transfer a disk from tower 3 to tower 1'.
transfer(m32,ToText):-
ToText = 'Transfer a disk from tower 3 to tower 2'.
```

Demo

```
?- transfer_discs.
First display_sequence ...
Transfer a disk from tower 3 to tower 1
Transfer a disk from tower 1 to tower 2
Transfer a disk from tower 1 to tower 3
Transfer a disk from tower 2 to tower 1
Second display_sequence ...
Transfer a disk from tower 1 to tower 3
Transfer a disk from tower 1 to tower 2
Transfer a disk from tower 3 to tower 2
Transfer a disk from tower 1 to tower 3
Transfer a disk from tower 2 to tower 1
Transfer a disk from tower 2 to tower 3
Transfer a disk from tower 1 to tower 3
true.
```

Task 7

```
Move = m13
NextState = [[m,l],[t],[s]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]]]
Move = m12
NextState = [[l],[m,t],[s]]
Move = m13
NextState = [[l],[t],[m,s]]
Move = m21
NextState = [[t,m,l],[],[s]]
Move = m23
NextState = [[m,l],[],[t,s]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]]]
Move = m12
NextState = [[l],[m],[t,s]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]]]
Move = m12
NextState = [[],[l,m],[t,s]]
Move = m13
NextState = [[],[m],[l,t,s]]
Move = m21
NextState = [[m,l],[],[t,s]]
Move = m23
NextState = [[l],[],[m,t,s]]
Move = m31
NextState = [[t,l],[m],[s]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]]]
Move = m12
NextState = [[l],[t,m],[s]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]]]
Move = m12
NextState = [[],[l,t,m],[s]]
Move = m13
NextState = [[],[t,m],[l,s]]
Move = m21
NextState = [[t,l],[m],[s]]
Move = m23
NextState = [[l],[m],[t,s]]
Move = m31
NextState = [[s,l],[t,m],[]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]]]
Move = m12
NextState = [[l],[s,t,m],[]]
Move = m13

Move = m21
NextState = [[t,s,l],[m],[]]
Move = m23
NextState = [[s,l],[m],[t]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]]]
Move = m12
NextState = [[l],[s,m],[t]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l
],[s,m],[t]]]
Move = m12
NextState = [[],[l,s,m],[t]]
Move = m13
NextState = [[],[s,m],[l,t]]
Move = m21
NextState = [[s,l],[m],[t]]
Move = m23
NextState = [[l],[m],[s,t]]
Move = m31
NextState = [[t,l],[s,m],[]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l
],[s,m],[t]],[[t,l],[s,m],[]]]
Move = m12
NextState = [[l],[t,s,m],[]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l
],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]]]
Move = m12
NextState = [[],[l,t,s,m],[]]
Move = m13
NextState = [[],[t,s,m],[l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l
],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]]]
Move = m21
NextState = [[t],[s,m],[l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l
],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]]]
Move = m12
NextState = [[],[t,s,m],[l]]
Move = m13
NextState = [[],[s,m],[t,l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l
],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]],[[],[s,m],[t,l]]]
Move = m21
NextState = [[s],[m],[t,l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l
],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]],[[],[s,m],[t,l]],[[s],[m],[t,l]]]
Move = m12
NextState = [[],[s,m],[t,l]]
Move = m13
NextState = [[],[m],[s,t,l]]
```

Move = m21
NextState = [[m,s],[],[t,l]]
Move = m23
NextState = [[s],[],[m,t,l]]
Move = m31
NextState = [[t,s],[m],[l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]],[[],[s,m],[t,l]],[[s],[m],[t,l]],[[t,s],[m],[l]]]
Move = m12
NextState = [[s],[t,m],[l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]],[[],[s,m],[t,l]],[[s],[m],[t,l]],[[t,s],[m],[l]],[[s],[t,m],[l]]]
Move = m12
NextState = [[],[s,t,m],[l]]
Move = m13
NextState = [[],[t,m],[s,l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]],[[],[s,m],[t,l]],[[s],[m],[t,l]],[[t,s],[m],[l]],[[s],[t,m],[l]],[[],[t,m],[s,l]]]
Move = m21
NextState = [[t],[m],[s,l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]],[[],[s,m],[t,l]],[[s],[m],[t,l]],[[t,s],[m],[l]],[[s],[t,m],[l]],[[],[t,m],[s,l]],[[t],[m],[s,l]]]
Move = m12
NextState = [[],[t,m],[s,l]]
Move = m13
NextState = [[],[m],[t,s,l]]
Move = m21
NextState = [[m,t],[],[s,l]]
Move = m23
NextState = [[t],[],[m,s,l]]
Move = m31
NextState = [[s,t],[m],[l]]
Move = m32
NextState = [[t],[s,m],[l]]
Move = m23
NextState = [[],[m],[t,s,l]]
Move = m31
NextState = [[s],[t,m],[l]]
Move = m32
NextState = [[],[s,t,m],[l]]
Move = m21
NextState = [[t,s],[m],[l]]
Move = m23
NextState = [[s],[m],[t,l]]
Move = m31
NextState = [[l,s],[t,m],[]]
Move = m32
NextState = [[s],[l,t,m],[]]
Move = m13
NextState = [[s],[m],[t,l]]

```
Move = m21
NextState = [[t,s],[m],[l]]
Move = m23
NextState = [[s],[m],[t,l]]
Move = m31
NextState = [[l,s],[t,m],[]]
Move = m32
NextState = [[s],[l,t,m],[]]
Move = m13
NextState = [[s],[m],[t,l]]
Move = m21
NextState = [[m,t,s],[],[l]]
Move = m23
NextState = [[t,s],[],[m,l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]],[[],[s,m],[t,l]],[[s],[m],[t,l]],[[t,s],[m],[l]],[[t,s],[],[m,l]]]
Move = m12
NextState = [[s],[t],[m,l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]],[[],[s,m],[t,l]],[[s],[m],[t,l]],[[t,s],[m],[l]],[[t,s],[],[m,l]],[[s],[t],[m,l]]]
Move = m12
NextState = [[],[s,t],[m,l]]
Move = m13
NextState = [[],[t],[s,m,l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]],[[],[s,m],[t,l]],[[s],[m],[t,l]],[[t,s],[m],[l]],[[t,s],[],[m,l]],[[s],[t],[m,l]],[[],[t],[s,m,l]]]
Move = m21
NextState = [[t],[],[s,m,l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]],[[],[s,m],[t,l]],[[s],[m],[t,l]],[[t,s],[m],[l]],[[t,s],[],[m,l]],[[s],[t],[m,l]],[[],[t],[s,m,l]],[[t],[],[s,m,l]]]
Move = m12
NextState = [[],[t],[s,m,l]]
Move = m13
NextState = [[],[],[t,s,m,l]]
PathSoFar = [[[t,s,m,l],[],[]],[[s,m,l],[t],[]],[[m,l],[t],[s]],[[m,l],[],[t,s]],[[l],[m],[t,s]],[[t,l],[m],[s]],[[l],[t,m],[s]],[[s,l],[t,m],[]],[[s,l],[m],[t]],[[l],[s,m],[t]],[[t,l],[s,m],[]],[[l],[t,s,m],[]],[[],[t,s,m],[l]],[[t],[s,m],[l]],[[],[s,m],[t,l]],[[s],[m],[t,l]],[[t,s],[m],[l]],[[t,s],[],[m,l]],[[s],[t],[m,l]],[[],[t],[s,m,l]],[[t],[],[s,m,l]],[[],[],[t,s,m,l]]]
SolutionSoFar = [m12,m13,m23,m12,m31,m12,m31,m23,m12,m31,m12,m13,m21,m13,m21,m31,m23,m12,m13,m21,m13]
```


1. What was the length of the solution? 14
2. What is the length of the shortest solution? 7
3. How do you account for the discrepancy?

Because it is a set algorithm, it will work the same every time.

Task 8

```
Solution ...

Transfer a disk from tower 1 to tower 2
Transfer a disk from tower 1 to tower 3
Transfer a disk from tower 2 to tower 3
Transfer a disk from tower 1 to tower 2
Transfer a disk from tower 3 to tower 1
Transfer a disk from tower 1 to tower 2
Transfer a disk from tower 3 to tower 1
Transfer a disk from tower 2 to tower 3
Transfer a disk from tower 1 to tower 2
Transfer a disk from tower 3 to tower 1
Transfer a disk from tower 1 to tower 2
Transfer a disk from tower 1 to tower 3
Transfer a disk from tower 2 to tower 1
Transfer a disk from tower 1 to tower 3
Transfer a disk from tower 2 to tower 1
Transfer a disk from tower 3 to tower 1
Transfer a disk from tower 2 to tower 3
Transfer a disk from tower 1 to tower 2
Transfer a disk from tower 1 to tower 3
Transfer a disk from tower 2 to tower 1
Transfer a disk from tower 1 to tower 3

true
```

Task 9

```
m12([Tower1Before,Tower2Before,Tower3],[Tower1After,Tower2After,Tower3]) :-
Tower1Before = [H|T],
Tower1After = T,
Tower2Before = L,
Tower2After = [H|L].

m13([Tower1Before,Tower2,Tower3Before],[Tower1After,Tower2,Tower3After]) :-
Tower1Before = [H|T],
Tower1After = T,
Tower3Before = L,
Tower3After = [H|L].

m21([Tower1Before,Tower2Before,Tower3],[Tower1After,Tower2After,Tower3]) :-
Tower2Before = [H|T],
Tower2After = T,
Tower1Before = L,
Tower1After = [H|L].

m23([Tower1,Tower2Before,Tower3Before],[Tower1,Tower2After,Tower3After]) :-
Tower2Before = [H|T],
Tower2After = T,
Tower3Before = L,
Tower3After = [H|L].

m31([Tower1Before,Tower2,Tower3Before],[Tower1After,Tower2,Tower3After]) :-
Tower3Before = [H|T],
Tower3After = T,
Tower1Before = L,
Tower1After = [H|L].

m32([Tower1,Tower2Before,Tower3Before],[Tower1,Tower2After,Tower3After]) :-
Tower3Before = [H|T],
Tower3After = T,
Tower2Before = L,
Tower2After = [H|L].
```

```
show(Name,Value) :-
write(Name),write(' = '),
write(Value),nl.

showr(Name,Value) :-
write(Name),write(' = '),
reverse(Value,RValue),
write(RValue),nl.

 check(Label,Name,Value) :-
write(Label),
write(Name),write(' = '),
write(Value),nl,
read(_).

checkr(Label,Name,Value) :-
write(Label),
write(Name),write(' = '),
reverse(Value,RValue),
write(RValue),nl,
read(_).

trace(Label,Name,Value) :-
write(Label),
write(Name),write(' = '),
write(Value),nl.
```

```
test__m12 :-
write('Testing: move_m12\n'),
TowersBefore = [[t,s,m,l,h],[],[]],
trace('','TowersBefore',TowersBefore),
m12(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).

test__m13 :-
write('Testing: move_m13\n'),
TowersBefore = [[t,s,m,l,h],[],[]],
trace('','TowersBefore',TowersBefore),
m13(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).

test__m21 :-
write('Testing: move_m21\n'),
TowersBefore = [[],[t,s,m,l,h],[]],
trace('','TowersBefore',TowersBefore),
m21(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).

test__m23 :-
write('Testing: move_m23\n'),
TowersBefore = [[],[t,s,m,l,h],[]],
trace('','TowersBefore',TowersBefore),
m23(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).

test__m31 :-
write('Testing: move_m31\n'),
TowersBefore = [[],[],[t,s,m,l,h]],
trace('','TowersBefore',TowersBefore),
m31(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).
```

```
test__m32 :-
write('Testing: move_m32\n'),
TowersBefore = [[],[],[t,s,m,l,h]],
trace('','TowersBefore',TowersBefore),
m32(TowersBefore,TowersAfter),
trace('','TowersAfter',TowersAfter).
```

```
disc_position([P1,P2,P3]) :-
correctspot(P1),
correctspot(P2),
correctspot(P3).
correctspot([]).
correctspot([t]).
correctspot([t,s]).
correctspot([t,m]).
correctspot([t,l]).
correctspot([t,h]).
correctspot([t,s,m]).
correctspot([t,s,m,l]).
correctspot([t,s,m,h]).
correctspot([t,s,m,l,h]).
correctspot([s]).
correctspot([s,m]).
correctspot([s,l]).
correctspot([s,h]).
correctspot([s,m,l]).
correctspot([s,m,h]).
correctspot([s,m,l,h]).
correctspot([m]).
correctspot([m,l]).
correctspot([m,h]).
correctspot([m,l,h]).
correctspot([l]).
correctspot([l,h]).
correctspot([h]).
```

```
 test_production :-
 write('Testing: production\n'),
 test_v([[l,t,s,m,h],[],[]]),
 test_v([[t,s,m,l,h],[],[]]),
 test_v([[],[h,t,s,m],[l]]),
 test_v([[],[t,s,m,h],[l]]),
 test_v([[],[h],[l,m,s,t]]),
 test_v([[],[h],[t,s,m,l]]).

test_v(V) :-
disc_position(V),
write(V), write(' is valid.'), nl.

test_v(V) :-
write(V), write(' is invalid.'), nl.

display_sequence([]).

display_sequence([H|T]) :-
transfer(H,E),
write(E),nl,
display_sequence(T).


transfer(m12,ToText):-
ToText = 'Transfer a disk from tower 1 to tower 2'.
transfer(m13,ToText):-
ToText = 'Transfer a disk from tower 1 to tower 3'.
transfer(m21,ToText):-
ToText = 'Transfer a disk from tower 2 to tower 1'.
transfer(m23,ToText):-
ToText = 'Transfer a disk from tower 2 to tower 3'.
transfer(m31,ToText):-
ToText = 'Transfer a disk from tower 3 to tower 1'.
transfer(m32,ToText):-
ToText = 'Transfer a disk from tower 3 to tower 2'.


transfer_discs :-
write('First display_sequence ...'), nl,
display_sequence([m31,m12,m13,m21]),
write('Second display_sequence ...'), nl,
display_sequence([m13,m12,m32,m13,m21,m23,m13]).
```

```
make_move(TowersBeforeMove,TowersAfterMove,m12) :-
m12(TowersBeforeMove,TowersAfterMove).
make_move(TowersBeforeMove,TowersAfterMove,m13) :-
m13(TowersBeforeMove,TowersAfterMove).
make_move(TowersBeforeMove,TowersAfterMove,m21) :-
m21(TowersBeforeMove,TowersAfterMove).
make_move(TowersBeforeMove,TowersAfterMove,m23) :-
m23(TowersBeforeMove,TowersAfterMove).
make_move(TowersBeforeMove,TowersAfterMove,m31) :-
m31(TowersBeforeMove,TowersAfterMove).
make_move(TowersBeforeMove,TowersAfterMove,m32) :-
m32(TowersBeforeMove,TowersAfterMove).


solution :-
extend_path([[[t,s,m,l],[],[]]],[],Solution),
write_solution(Solution).

extend_path(PathSoFar,SolutionSoFar,Solution):-
PathSoFar = [[[],[],[t,s,m,l]]|_],
showr('PathSoFar',PathSoFar),
showr('SolutionSoFar',SolutionSoFar),
Solution = SolutionSoFar.

extend_path(PathSoFar,SolutionSoFar,Solution):-
PathSoFar = [CurrentState|_],
showr('PathSoFar',PathSoFar),
make_move(CurrentState,NextState,Move),
show('Move',Move),
show('NextState',NextState),
not(member(NextState,PathSoFar)),
disc_position(NextState),
Path = [NextState|PathSoFar],
Soln = [Move|SolutionSoFar],
extend_path(Path,Soln,Solution).

write_solution(S) :-
nl, write('Solution:'), nl, nl,
reverse(S,R),
display_sequence(R),nl.
```