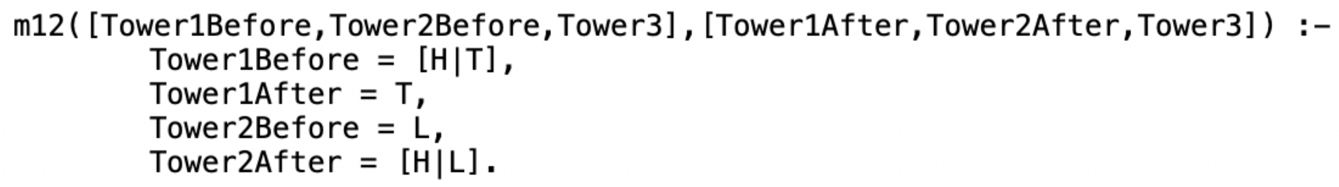
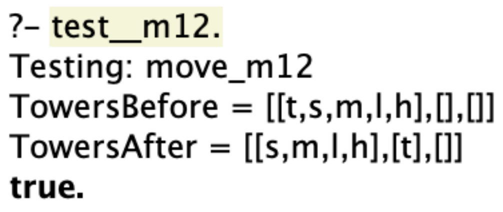
Second Prolog Programming Assignment Solution

Task 3: One Move Predicate and a Unit Test



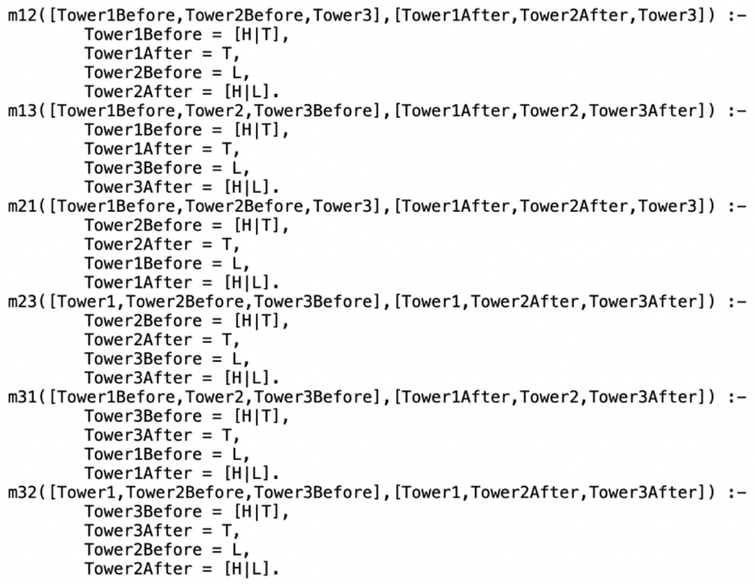
```
m12([Tower1Before,Tower2Before,Tower3],[Tower1After,Tower2After,Tower3]) :-
        Tower1Before = [H|T],
        Tower1After = T,
        Tower2Before = L,
        Tower2After = [H|L].
```

Unit Test Demo



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

Task 4: The Remaining Five Move Predicates and a Unit Tests



```
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].
```

*

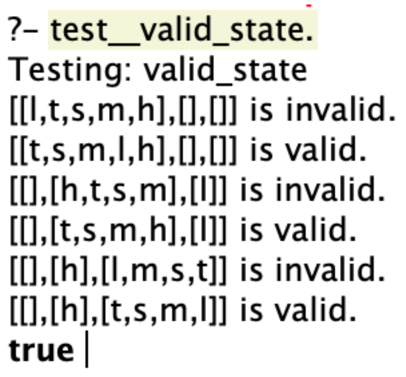
Unit Test Demo

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

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

?- 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.
```

Task 5: Valid State Predicate and Unit Test

```
valid_state([Tower1,Tower2,Tower3]) :-
        valid_tower(Tower1),
        valid_tower(Tower2),
        valid_tower(Tower3).

valid_tower([]).

valid_tower([t]).
valid_tower([s]).
valid_tower([m]).
valid_tower([l]).
valid_tower([h]).

valid_tower([t, s]).
valid_tower([t, m]).
valid_tower([t, l]).
valid_tower([t, h]).

valid_tower([s, m]).
valid_tower([s, l]).
valid_tower([s, h]).

valid_tower([m, h]).
valid_tower([m, l]).

valid_tower([l, h]).


valid_tower([t, s, h]).
valid_tower([t, s, l]).
valid_tower([t, s, m]).
valid_tower([t, m, h]).
valid_tower([t, m, l]).

valid_tower([s, m, l]).
valid_tower([s, m, h]).
valid_tower([s, l, h]).

valid_tower([m, l, h]).

valid_tower([t, s, l, h]).
valid_tower([t, m, l, h]).
valid_tower([t, s, m, h]).
valid_tower([t, s, m, l]).

valid_tower([s, m, l, h]).

valid_tower([t, s, m, l, h]).
```

Unit Test Program Demo

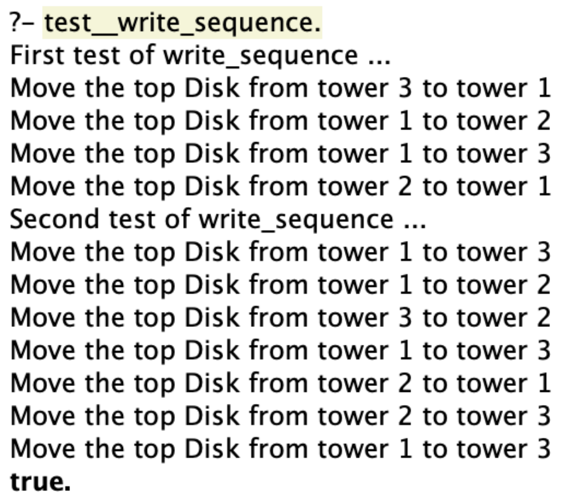


```
?- test__valid_state.
Testing: valid_state
[[l,t,s,m,h],[],[]] is invalid.
[[t,s,m,l,h],[],[]] is valid.
[[],[h,t,s,m],[l]] is invalid.
[[],[t,s,m,h],[l]] is valid.
[[],[h],[l,m,s,t]] is invalid.
[[],[h],[t,s,m,l]] is valid.
true
```

Task 6: Defining the write sequence predicate

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

write_sequence([]).

write_sequence([H|T]) :-
        elaborate(H, E), write(E), nl, write_sequence(T).

elaborate(m12, E) :-
        E = "Move the top Disk from tower 1 to tower 2.".
elaborate(m13, E) :-
        E = "Move the top Disk from tower 1 to tower 3.".
elaborate(m21, E) :-
        E = "Move the top Disk from tower 2 to tower 1.".
elaborate(m23, E) :-
        E = "Move the top Disk from tower 2 to tower 3.".
elaborate(m31, E) :-
        E = "Move the top Disk from tower 3 to tower 1.".
elaborate(m32, E) :-
        E = "Move the top Disk from tower 3 to tower 2.".
```

Unit Test Program Demo



```
?- test__write_sequence.
First test of write_sequence ...
Move the top Disk from tower 3 to tower 1
Move the top Disk from tower 1 to tower 2
Move the top Disk from tower 1 to tower 3
Move the top Disk from tower 2 to tower 1
Second test of write_sequence ...
Move the top Disk from tower 1 to tower 3
Move the top Disk from tower 1 to tower 2
Move the top Disk from tower 3 to tower 2
Move the top Disk from tower 1 to tower 3
Move the top Disk from tower 2 to tower 1
Move the top Disk from tower 2 to tower 3
Move the top Disk from tower 1 to tower 3
true.
```

Task 7: Run the program to solve the 3 disk problem

```
?- solve.

PathSoFar = [[[s,m,l],[],[]]]

Move = m12

NextState = [[m,l],[s],[]]

PathSoFar = [[[s,m,l],[],[]],[[m,l],[s],[]]]

Move = m12

NextState = [[l],[m,s],[]]

Move = m13

NextState = [[l],[s],[m]]

PathSoFar = [[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]]]

Move = m12

NextState = [[],[l,s],[m]]

Move = m13

NextState = [[],[s],[l,m]]

Move = m21

NextState = [[s,l],[],[m]]

PathSoFar = [[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]]]

Move = m12

NextState = [[l],[s],[m]]

Move = m13

NextState = [[l],[],[s,m]]

PathSoFar = [[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]]]

Move = m12

NextState = [[],[l],[s,m]]

PathSoFar = [[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]]]

Move = m21

NextState = [[l],[],[s,m]]

Move = m23

NextState = [[],[],[l,s,m]]

Move = m31

NextState = [[s],[l],[m]]

PathSoFar = [[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]]]

Move = m12
```

```
NextState = [[],[s,l],[m]]

PathSoFar = [[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]]]

Move = m21

NextState = [[s],[l],[m]]

Move = m23

NextState = [[],[l],[s,m]]

Move = m31

NextState = [[m],[s,l],[]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]]]

Move = m12

NextState = [[],[m,s,l],[]]

Move = m13

NextState = [[],[s,l],[m]]

Move = m21

NextState = [[s,m],[l],[]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]]]

Move = m12

NextState = [[m],[s,l],[]]

Move = m13

NextState = [[m],[l],[s]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]],[[m],[l],[s]]]

Move = m12

NextState = [[],[m,l],[s]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]],[[m],[l],[s]],[[],[m,l],[s]]]

Move = m21

NextState = [[m],[l],[s]]

Move = m23

NextState = [[],[l],[m,s]]

Move = m31
```

```
NextState = [[s],[m,l],[]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]],[[m],[l],[s]],[[],[m,l],[s]],[[s],[m,l],[]]]

Move = m12

NextState = [[],[s,m,l],[]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]],[[m],[l],[s]],[[],[m,l],[s]],[[s],[m,l],[]],[[],[s,m,l],[]]]

Move = m21

NextState = [[s],[m,l],[]]

Move = m23

NextState = [[],[m,l],[s]]

Move = m13

NextState = [[],[m,l],[s]]

Move = m21

NextState = [[m,s],[l],[]]

Move = m23

NextState = [[s],[l],[m]]

Move = m32

NextState = [[],[s,m,l],[]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]],[[m],[l],[s]],[[],[m,l],[s]],[[],[s,m,l],[]]]

Move = m21

NextState = [[s],[m,l],[]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]],[[m],[l],[s]],[[],[m,l],[s]],[[],[s,m,l],[]],[[s],[m,l],[]]]

Move = m12

NextState = [[],[s,m,l],[]]

Move = m13

NextState = [[],[m,l],[s]]

Move = m21

NextState = [[m,s],[l],[]]

Move = m23
```

```
NextState = [[s],[l],[m]]

Move = m23

NextState = [[],[m,l],[s]]

Move = m13

NextState = [[],[l],[m,s]]

Move = m21

NextState = [[l,m],[],[s]]

Move = m23

NextState = [[m],[],[l,s]]

Move = m31

NextState = [[s,m],[l],[]]

Move = m32

NextState = [[m],[s,l],[]]

Move = m21

NextState = [[l,s,m],[],[]]

Move = m23

NextState = [[s,m],[],[l]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]],[[s,m],[],[l]]]

Move = m12

NextState = [[m],[s],[l]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]],[[s,m],[],[l]],[[m],[s],[l]]]

Move = m12

NextState = [[],[m,s],[l]]

Move = m13

NextState = [[],[s],[m,l]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]],[[s,m],[],[l]],[[m],[s],[l]],[[],[s],[m,l]]]

Move = m21

NextState = [[s],[],[m,l]]
```

```
PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]],[[s,m],[],[l]],[[m],[s],[l]],[[],[s],[m,l]],[[s],[],[m,l]]]

Move = m12

NextState = [[],[s],[m,l]]

Move = m13

NextState = [[],[],[s,m,l]]

PathSoFar =
[[[s,m,l],[],[]],[[m,l],[s],[]],[[l],[s],[m]],[[s,l],[],[m]],[[l],[],[s,m]],[[],[l],[s,m]],[[s],[l],[m]],[[],[s,l],[m]],[[m],[s,l],[]],[[s,
m],[l],[]],[[s,m],[],[l]],[[m],[s],[l]],[[],[s],[m,l]],[[s],[],[m,l]],[[],[],[s,m,l]]]

SolutionSoFar = [m12,m13,m21,m13,m12,m31,m12,m31,m21,m23,m12,m13,m21,m13]

Solution ...

Move the top Disk from tower 1 to tower 2.

Move the top Disk from tower 1 to tower 3.

Move the top Disk from tower 2 to tower 1.

Move the top Disk from tower 1 to tower 3.

Move the top Disk from tower 1 to tower 2.

Move the top Disk from tower 3 to tower 1.

Move the top Disk from tower 1 to tower 2.

Move the top Disk from tower 3 to tower 1.

Move the top Disk from tower 2 to tower 1.

Move the top Disk from tower 2 to tower 3.

Move the top Disk from tower 1 to tower 2.

Move the top Disk from tower 1 to tower 3.

Move the top Disk from tower 2 to tower 1.

Move the top Disk from tower 1 to tower 3.

true
```

Task 8: Run the program to solve the 4 disk problem

```
?- solve.

PathSoFar = [[[s,m,l,h],[],[]]]

Move = m12

NextState = [[m,l,h],[s],[]]

PathSoFar = [[[s,m,l,h],[],[]],[[m,l,h],[s],[]]]
```

```
Move = m12

NextState = [[l,h],[m,s],[]]

Move = m13

NextState = [[l,h],[s],[m]]

PathSoFar = [[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]]]

Move = m12

NextState = [[h],[l,s],[m]]

Move = m13

NextState = [[h],[s],[l,m]]

Move = m21

NextState = [[s,l,h],[],[m]]

PathSoFar = [[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]]]

Move = m12

NextState = [[l,h],[s],[m]]

Move = m13

NextState = [[l,h],[],[s,m]]

PathSoFar = [[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]]]

Move = m12

NextState = [[h],[l],[s,m]]

PathSoFar = [[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]]]

Move = m12

NextState = [[],[h,l],[s,m]]

Move = m13

NextState = [[],[l],[h,s,m]]

Move = m21

NextState = [[l,h],[],[s,m]]

Move = m23

NextState = [[h],[],[l,s,m]]

Move = m31

NextState = [[s,h],[l],[m]]

PathSoFar = [[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]]]

Move = m12

NextState = [[h],[s,l],[m]]
```

```
PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]]]

Move = m12

NextState = [[],[h,s,l],[m]]

Move = m13

NextState = [[],[s,l],[h,m]]

Move = m21

NextState = [[s,h],[l],[m]]

Move = m23

NextState = [[h],[l],[s,m]]

Move = m31

NextState = [[m,h],[s,l],[]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]]]

Move = m12

NextState = [[h],[m,s,l],[]]

Move = m13

NextState = [[h],[s,l],[m]]

Move = m21

NextState = [[s,m,h],[l],[]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]]]

Move = m12

NextState = [[m,h],[s,l],[]]

Move = m13

NextState = [[m,h],[l],[s]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]]]

Move = m12

NextState = [[h],[m,l],[s]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]]]

Move = m12
```

```
NextState = [[],[h,m,l],[s]]

Move = m13

NextState = [[],[m,l],[h,s]]

Move = m21

NextState = [[m,h],[l],[s]]

Move = m23

NextState = [[h],[l],[m,s]]

Move = m31

NextState = [[s,h],[m,l],[]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]]]

Move = m12

NextState = [[h],[s,m,l],[]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]]]

Move = m12

NextState = [[],[h,s,m,l],[]]

Move = m13

NextState = [[],[s,m,l],[h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]]]

Move = m21

NextState = [[s],[m,l],[h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]]]

Move = m12

NextState = [[],[s,m,l],[h]]

Move = m13

NextState = [[],[m,l],[s,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]]]
```

```
Move = m21

NextState = [[m],[l],[s,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]]]

Move = m12

NextState = [[],[m,l],[s,h]]

Move = m13

NextState = [[],[l],[m,s,h]]

Move = m21

NextState = [[l,m],[],[s,h]]

Move = m23

NextState = [[m],[],[l,s,h]]

Move = m31

NextState = [[s,m],[l],[h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]]]

Move = m12

NextState = [[m],[s,l],[h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]]]

Move = m12

NextState = [[],[m,s,l],[h]]

Move = m13

NextState = [[],[s,l],[m,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]]]

Move = m21

NextState = [[s],[l],[m,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]]]

Move = m12

NextState = [[],[s,l],[m,h]]

Move = m13

NextState = [[],[l],[s,m,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]]]

Move = m21

NextState = [[l],[],[s,m,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]]]

Move = m12

NextState = [[],[l],[s,m,h]]

Move = m13

NextState = [[],[],[l,s,m,h]]

Move = m31

NextState = [[s,l],[],[m,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]]]

Move = m12

NextState = [[l],[s],[m,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]]]

Move = m12

NextState = [[],[l,s],[m,h]]

Move = m13

NextState = [[],[s],[l,m,h]]

Move = m21
```

```
NextState = [[s,l],[],[m,h]]

Move = m23

NextState = [[l],[],[s,m,h]]

Move = m31

NextState = [[m,l],[s],[h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]]]

Move = m12

NextState = [[l],[m,s],[h]]

Move = m13

NextState = [[l],[s],[m,h]]

Move = m21

NextState = [[s,m,l],[],[h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]]]

Move = m12

NextState = [[m,l],[s],[h]]

Move = m13

NextState = [[m,l],[],[s,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]]]

Move = m12

NextState = [[l],[m],[s,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]]]

Move = m12

NextState = [[],[l,m],[s,h]]
```

```
Move = m13

NextState = [[],[m],[l,s,h]]

Move = m21

NextState = [[m,l],[],[s,h]]

Move = m23

NextState = [[l],[],[m,s,h]]

Move = m31

NextState = [[s,l],[m],[h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]],[[s,l],[m],[h]]]

Move = m12

NextState = [[l],[s,m],[h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]],[[s,l],[m],[h]],[[l],[s,m],[h]]]

Move = m12

NextState = [[],[l,s,m],[h]]

Move = m13

NextState = [[],[s,m],[l,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]],[[s,l],[m],[h]],[[l],[s,m],[h]],[[],[s,m],[l,h
]]]

Move = m21

NextState = [[s],[m],[l,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]],[[s,l],[m],[h]],[[l],[s,m],[h]],[[],[s,m],[l,h
]],[[s],[m],[l,h]]]

Move = m12

NextState = [[],[s,m],[l,h]]
```

```
Move = m13

NextState = [[],[m],[s,l,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]],[[s,l],[m],[h]],[[l],[s,m],[h]],[[],[s,m],[l,h
]],[[s],[m],[l,h]],[[],[m],[s,l,h]]]

Move = m21

NextState = [[m],[],[s,l,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]],[[s,l],[m],[h]],[[l],[s,m],[h]],[[],[s,m],[l,h
]],[[s],[m],[l,h]],[[],[m],[s,l,h]],[[m],[],[s,l,h]]]

Move = m12

NextState = [[],[m],[s,l,h]]

Move = m13

NextState = [[],[],[m,s,l,h]]

Move = m31

NextState = [[s,m],[],[l,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]],[[s,l],[m],[h]],[[l],[s,m],[h]],[[],[s,m],[l,h
]],[[s],[m],[l,h]],[[],[m],[s,l,h]],[[m],[],[s,l,h]],[[s,m],[],[l,h]]]

Move = m12

NextState = [[m],[s],[l,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]],[[s,l],[m],[h]],[[l],[s,m],[h]],[[],[s,m],[l,h
]],[[s],[m],[l,h]],[[],[m],[s,l,h]],[[m],[],[s,l,h]],[[s,m],[],[l,h]],[[m],[s],[l,h]]]

Move = m12

NextState = [[],[m,s],[l,h]]

Move = m13

NextState = [[],[s],[m,l,h]]
```

```
PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]],[[s,l],[m],[h]],[[l],[s,m],[h]],[[],[s,m],[l,h
]],[[s],[m],[l,h]],[[],[m],[s,l,h]],[[m],[],[s,l,h]],[[s,m],[],[l,h]],[[m],[s],[l,h]],[[],[s],[m,l,h]]]

Move = m21

NextState = [[s],[],[m,l,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]],[[s,l],[m],[h]],[[l],[s,m],[h]],[[],[s,m],[l,h
]],[[s],[m],[l,h]],[[],[m],[s,l,h]],[[m],[],[s,l,h]],[[s,m],[],[l,h]],[[m],[s],[l,h]],[[],[s],[m,l,h]],[[s],[],[m,l,h]]]

Move = m12

NextState = [[],[s],[m,l,h]]

Move = m13

NextState = [[],[],[s,m,l,h]]

PathSoFar =
[[[s,m,l,h],[],[]],[[m,l,h],[s],[]],[[l,h],[s],[m]],[[s,l,h],[],[m]],[[l,h],[],[s,m]],[[h],[l],[s,m]],[[s,h],[l],[m]],[[h],[s,l],[m]],[[
m,h],[s,l],[]],[[s,m,h],[l],[]],[[m,h],[l],[s]],[[h],[m,l],[s]],[[s,h],[m,l],[]],[[h],[s,m,l],[]],[[],[s,m,l],[h]],[[s],[m,l],[h]],[[],[
m,l],[s,h]],[[m],[l],[s,h]],[[s,m],[l],[h]],[[m],[s,l],[h]],[[],[s,l],[m,h]],[[s],[l],[m,h]],[[],[l],[s,m,h]],[[l],[],[s,m,h]],[[s,l],[],
[m,h]],[[l],[s],[m,h]],[[m,l],[s],[h]],[[s,m,l],[],[h]],[[m,l],[],[s,h]],[[l],[m],[s,h]],[[s,l],[m],[h]],[[l],[s,m],[h]],[[],[s,m],[l,h
]],[[s],[m],[l,h]],[[],[m],[s,l,h]],[[m],[],[s,l,h]],[[s,m],[],[l,h]],[[m],[s],[l,h]],[[],[s],[m,l,h]],[[s],[],[m,l,h]],[[],[],[s,m,l,h]]]

SolutionSoFar =
[m12,m13,m21,m13,m12,m31,m12,m31,m21,m13,m12,m31,m12,m13,m21,m13,m21,m31,m12,m13,m21,m13,
m21,m31,m12,m31,m21,m13,m12,m31,m12,m13,m21,m13,m21,m31,m12,m13,m21,m13]


Solution ...


Move the top Disk from tower 1 to tower 2.

Move the top Disk from tower 1 to tower 3.

Move the top Disk from tower 2 to tower 1.

Move the top Disk from tower 1 to tower 3.

Move the top Disk from tower 1 to tower 2.

Move the top Disk from tower 3 to tower 1.

Move the top Disk from tower 1 to tower 2.

Move the top Disk from tower 3 to tower 1.

Move the top Disk from tower 2 to tower 1.

Move the top Disk from tower 1 to tower 3.
Move the top Disk from tower 1 to tower 2.
Move the top Disk from tower 3 to tower 1.
Move the top Disk from tower 1 to tower 2.
Move the top Disk from tower 1 to tower 3.
Move the top Disk from tower 2 to tower 1.
Move the top Disk from tower 1 to tower 3.
Move the top Disk from tower 2 to tower 1.
Move the top Disk from tower 3 to tower 1.
Move the top Disk from tower 1 to tower 2.
Move the top Disk from tower 1 to tower 3.
Move the top Disk from tower 2 to tower 1.
Move the top Disk from tower 1 to tower 3.
Move the top Disk from tower 2 to tower 1.
Move the top Disk from tower 3 to tower 1.
Move the top Disk from tower 1 to tower 2.
Move the top Disk from tower 3 to tower 1.
Move the top Disk from tower 2 to tower 1.
Move the top Disk from tower 1 to tower 3.
Move the top Disk from tower 1 to tower 2.
Move the top Disk from tower 3 to tower 1.
Move the top Disk from tower 1 to tower 2.
Move the top Disk from tower 1 to tower 3.
Move the top Disk from tower 2 to tower 1.
Move the top Disk from tower 1 to tower 3.
Move the top Disk from tower 2 to tower 1.
Move the top Disk from tower 3 to tower 1.
Move the top Disk from tower 1 to tower 2.
Move the top Disk from tower 1 to tower 3.
Move the top Disk from tower 2 to tower 1.
Move the top Disk from tower 1 to tower 3.
True
```

Task 9: Review your code and archive it

```
% ----------------------------------------------------------------------
% ----------------------------------------------------------------------
% --- File: towers_of_hanoi.pro
% --- Line: Program to solve the Towers of Hanoi problem
% ----------------------------------------------------------------------
:- consult('inspector.pro').
% ----------------------------------------------------------------------
% --- make_move(S,T,SSO) :: Make a move from state S to state T by SSO


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).


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].

% ---------------------------------------------------------------------
% --- valid_state(S) :: S is a valid state

valid_state([Tower1,Tower2,Tower3]) :-
valid_tower(Tower1),
valid_tower(Tower2),
valid_tower(Tower3).

valid_tower([]).

valid_tower([t]).
valid_tower([s]).
valid_tower([m]).
valid_tower([l]).
valid_tower([h]).

valid_tower([t, s]).
valid_tower([t, m]).
valid_tower([t, l]).
valid_tower([t, h]).

valid_tower([s, m]).
valid_tower([s, l]).
valid_tower([s, h]).
valid_tower([m, h]).
valid_tower([m, l]).

valid_tower([l, h]).



valid_tower([t, s, h]).
valid_tower([t, s, l]).
valid_tower([t, s, m]).
valid_tower([t, m, h]).
valid_tower([t, m, l]).

valid_tower([s, m, l]).
valid_tower([s, m, h]).
valid_tower([s, l, h]).

valid_tower([m, l, h]).

valid_tower([t, s, l, h]).
valid_tower([t, m, l, h]).
valid_tower([t, s, m, h]).
valid_tower([t, s, m, l]).

valid_tower([s, m, l, h]).

valid_tower([t, s, m, l, h]).

% ----------------------------------------------------------------------
% --- solve(Start,Solution) :: succeeds if Solution represents a path
% --- from the start state to the goal state.
solve :-
extend_path([[[s,m,l],[],[]]],[],Solution),
write_solution(Solution).
extend_path(PathSoFar,SolutionSoFar,Solution) :-
PathSoFar = [[[],[],[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)),
valid_state(NextState),
Path = [NextState|PathSoFar],
Soln = [Move|SolutionSoFar],
extend_path(Path,Soln,Solution).
% ---------------------------------------------------------------------
% --- write_sequence_reversed(S) :: Write the sequence, given by S,
% --- expanding the tokens into meaningful strings.
write_solution(S) :-
nl, write('Solution ...'), nl, nl, reverse(S,R), write_sequence(R),nl.

write_sequence([]).

write_sequence([H|T]) :-
elaborate(H, E), write(E), nl, write_sequence(T).

elaborate(m12, E) :-
E = "Move the top Disk from tower 1 to tower 2.".
elaborate(m13, E) :-
E = "Move the top Disk from tower 1 to tower 3.".
elaborate(m21, E) :-
E = "Move the top Disk from tower 2 to tower 1.".
elaborate(m23, E) :-
E = "Move the top Disk from tower 2 to tower 3.".
elaborate(m31, E) :-
E = "Move the top Disk from tower 3 to tower 1.".
elaborate(m32, E) :-
E = "Move the top Disk from tower 3 to tower 2.".



% ---------------------------------------------------------------------
% --- Unit test programs

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).



test__valid_state :-
write('Testing: valid_state\n'),
test__vs([[l,t,s,m,h],[],[]]),
test__vs([[t,s,m,l,h],[],[]]),
test__vs([[],[h,t,s,m],[l]]),
test__vs([[],[t,s,m,h],[l]]),
test__vs([[],[h],[l,m,s,t]]),
test__vs([[],[h],[t,s,m,l]]).

test__vs(S) :-
valid_state(S),
write(S),
write(' is valid.'),
nl.

test__vs(S) :-
write(S),
write(' is invalid.'),
nl.

test__write_sequence :-
write('First test of write_sequence ...'),
nl,
write_sequence([m31,m12,m13,m21]),
write('Second test of write_sequence ...'),
nl,
write_sequence([m13,m12,m32,m13,m21,m23,m13]).
```