Name: Purushottam Tiwari
Roll no.: 19074029
Branch: CSE(IDD)
Course: CSE-241

<u>Exercise-3.1</u>

The problem of the farmer, lion, goat and grain(barley) can be solved using an approach similar to **Solution-4** for **Tic-Tac-Toe** problem: We search in a graph of possible states in order to reach the goal and maintain

<u>Data structure:</u>
(i). Use nodes to represent the different next possible states of the problem in a graph.
(ii). A list **L** to store the already visited state, in order to decide cotinuation or end(succcess or failure) of move.

<u>Algorithm:</u>
An state(node) can be represented as state(F, L, G, B), i.e. the state(position) of (F,L,G,B) according to that state. The below described algorithms is based on Dijkstra algorithm:
(i). At any state we look out for new possible states using a set of state generations rules described later on.
(ii). In order to move from an state **X** to state **Y**, we define a recursive rule to check the validity of our move. So the path to from state **X** to state **Y**, is:

```
    path(X, Y, L):- move(X, Z)
                    not(member(Z, L)) // check Z is not visited
                    path(Z, Y, [Z|L]) // add Z to the list
```

The detatil of above rule is: There exists a path from state **X**, to state **Y** if we can move from state **X** to state **Y** using an intermediary state **Z** with a special case that a path from state **Z** to state **Z(itself)** always exists.
(iii). This search terminates when the path generated leads to the the goal state(successful) or loops back to the start state(failed).

The new possible states generation can be done using following rule:

In the following representation:
opp(a, b) means a/b should swap
opp(b, a) means b/a should swap
Unsafe state are represented as:
unsafe(state(X,Y,Y,B)) :- opp(X,Y) // lion eats goat
unsafe(state(X,L,Y,Y)) :- opp(X,Y) // goat eats barley

(i). Farmer takes lion to other side:
move(state(X,X,G,B), state(Y,Y,G,B)) :- opp(X,Y), not (unsafe(state(Y,Y,G,B)))
(ii). Farmer takes goat to other side:
move(state(X,L,X,B), state(Y,L,Y,B)) :- opp(X,Y), not (unsafe(state(Y,L,Y,B)))
(iii). Farmer takes himself to other side:
move(state(X,L,G,X), state(Y,L,G,Y)) :- opp(X,Y), not (unsafe(state(Y,L,G,B)))

(iv). Farmer takes himself to other side:
move(state(X,L,G,B), state(Y,L,G,B)) :- opp(X,Y), not (unsafe(state(Y,L,G,B)))
Rule for breaking the search, i.e. deciding success or failure(backtrack):
move(state(F,L,G,B), state(F,L,G,B)):-writelist([‘ Backtrack from:’, F,L,G,B]),fail.

The solution discussed above similar to **solution-4** (swhich uses tree traversal in a similar manner) of the **Tic-Tac-Toe** problem discussed in the class, as