**Name:** Satyam Jaiswal **UID:** 2021600028 **Batch**: B **Date:** 22/08/2023

**EXP 1**: To create a simple reflex agent

**Problem:** In the missionaries and cannibals problem, three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries). The boat cannot cross the river by itself with no people on board.

The missionaries and cannibals problem is usually stated as follows. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place.

1- Implement it with simple If else

a. Formulate the problem precisely, making only those distinctions necessary to ensure a

valid solution. Draw a diagram of the complete state space.

b. Implement and solve the problem optimally using an appropriate search algorithm.

**Program:**

#include <iostream>

#include <set>

using namespace std;

struct State {

int missionariesLeft, cannibalsLeft, boatPosition;

};

bool isValid(const State& s) {

if (s.missionariesLeft < 0 || s.missionariesLeft > 3 ||

s.cannibalsLeft < 0 || s.cannibalsLeft > 3)

return false;

if (s.missionariesLeft > 0 && s.missionariesLeft < s.cannibalsLeft)

return false;

int missionariesRight = 3 - s.missionariesLeft;

int cannibalsRight = 3 - s.cannibalsLeft;

if (missionariesRight > 0 && missionariesRight < cannibalsRight)

return false;

return true;

}

bool isGoal(const State& s) {

return s.missionariesLeft == 0 && s.cannibalsLeft == 0 && s.boatPosition == 1;

}

void printState(const State& s) {

cout << "Left Bank: " <<s.missionariesLeft<< "M "<<s.cannibalsLeft<<"C "<<(s.boatPosition==1 ? "Boat":"")<<"\t\t\t\t";

cout <<"Right Bank: " <<3 - s.missionariesLeft<<"M "<<3 - s.cannibalsLeft<<"C "<<(s.boatPosition!=1 ? "Boat ":"") <<endl;

}

void solve(State s) {

if (isGoal(s)) {

return;

}

if (isValid(s)) {

printState(s);

for (int m = 0; m <= 2; m++) {

for (int c = 0; c <= 2; c++) {

if (m + c >= 1 && m + c <= 2) {

State nextState = s;

nextState.missionariesLeft -= m;

nextState.cannibalsLeft -= c;

nextState.boatPosition = 1 - nextState.boatPosition;

if (isValid(nextState)) {

solve(nextState);

}

}

}

}

}

}

int main() {

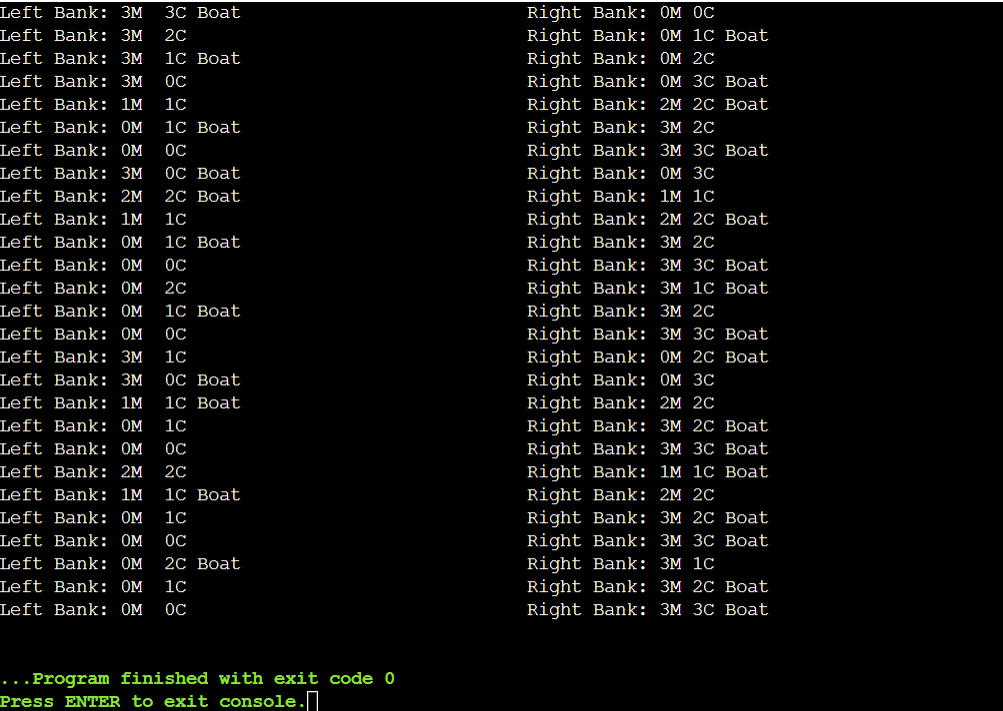
State initial = {3, 3, 1};

solve(initial);

return 0;

}

**Output:**

****