INDEX

NAME: Rulkit Shringi STD.: _____ SEC.: I I ROLL NO.: 596 SUB.: Artificial Intelligence

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1	22/1/24	Implementation of toy problem.	10	
2	31/1/24	Implementation of real world problem	10	31/01/24

classmate
Date 22/1/24
Page

Ex-1

Implementation of toy problem

- Aim:

To implement toy problems and understand its working algorithms

1) Camel and Banana Puzzle:

Conditions:

A person wants to transfer bananas over to a destination 'A' kms away. He initially has 'B' Bananas and the camel can carry upto 'c' bananas at a time. If the camel consumes one banana for every kilometre travelled, what is the maximum no. of bananas that can reach the destination.

- input: A = ~~total distance~~ (kms)
  B = total bananas
  C = Maximum bananas on a camel

output: Maximum bananas that can be transported

2) Missionaries and Cannibals:

Conditions:

In this problem 3 missionaries and 3 cannibals must cross a river on a boat which can carry at most 2 people. The boat cannot cross the river by itself and the cannibals should never outnumber the missionaries.

Input : M = no. of missionaries
C = no. of cannibals
B = maximum occupancy of the boat

output : If there exists a solution, the steps should be illustrated

3) N Puzzle Problem

Conditions :
You are given a grid of 'n x n' with 'n x n-1' tiles numbered 1 to 'n x n-1'. The task is to rearrange the randomly allotted numbers into a sorted order with the least no. of movements. We can slide the 4 adjacent sides of an empty tile into that tile. (left, right, above, below).

Input : N = size of grid (height)
output : Arrays of N x N showing the movement at each step till the solution is produced cost and number of moves is printed.

4) Water Jug Problem

Conditions :
There exists an 'm' liter jug and an 'n' liter Jug (m > n). Both are originally empty, and unmarked. You have to use the jugs to fill 'd' liters in the smaller jug (d ≤ n).

Operations :

1. Empty a jug : $(x, 0) \rightarrow (0, 0)$
2. Fill a jug : $(0, 0) \rightarrow (x, y)$
3. Pour water from one jug to another

$$(x, y) \rightarrow (x - d, y + d)$$

- Input : x = capacity of jug 1 (litres)
  y = capacity of jug 2 (litres)
  z = amount to be filled in jug (litres)

  output : steps to fill z litres in y.

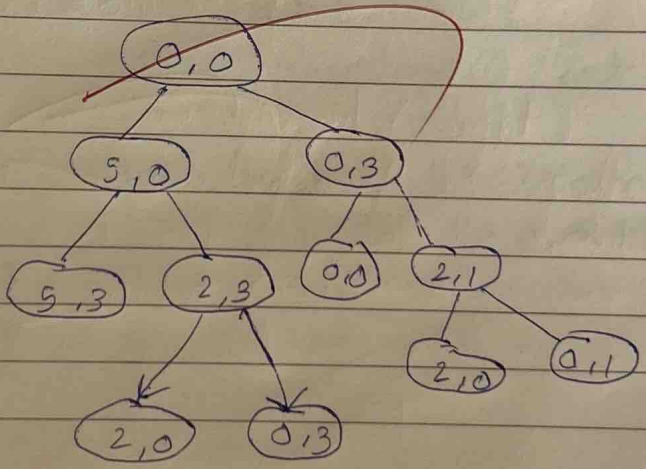
  Eg : $x = 4, y = 3, z = 2$ (input)

  output : {(0,0), (0,3), (3,0), (3,3), (4,2), (0,2)}

- Algorithm :

  State space graph :

  $x = 5, y = 3, z = 2$



Path cost = 3

• Program :-

```
#include <iostream>
#include <unordered_set>

using namespace std;

struct state {
    int x, y;
    bool operator==(const state& other)
    const {
        return x == other.x && y == other.y;
    }
};

namespace std {
    template<>
    struct hash<state> {
        size_t operator()(const state& s)
        const {
            return hash<int>()(s.x) ^ hash
            <int>()(s.y);
        }
    };
}

bool isGoalState(const state& state, int
target) {
    return state.x == target || state.y ==
    target;
}
```

```
void pourWater(int x, int y, int target, State
    current state, unordered_set<State>& visited)
{ if (isGoalState(current state, target)){
    cout << "To get " << target << " measure: "
    << current state.x << ", " << current state
    << ")" endl;
}
    visited.insert(current state);
    // x to y
if (current state.x > 0){
    int pourAmount = Min(current state.x, y -
                         current state.y);
State next state = {current state.x - pourAmt,
current state.y + pourAmt};

if (visited.find(next state) == visited.end()){
    cout << "Pour " << pourAmt << " x to y " <<
next state.x << ", " << next state.y << ")"
                                << endl;
pourWater(x, y, target, next state, visited);
}
}
// empt x
if (current State.x > 0){
    State next state = {0, current state.y};
if (visited.find(next state) == visited.end()){
& cout << "Empty x (" << next state.x << ", "
next state.y << ")" << endl;
pourWater(x, x, target, next state, visited);
}
}
```

Output

Enter capacity of jug x : 4
Enter capacity of jug y : 3
enter target amount : 2

Solution steps :
Fill x (4,0)
Pour 3 from x to y (1,3)
~~empty 3 x to x (3,0~~
empty x (0,3)
Pour 3 from to x (3,0)
Fill y (3,3)
Pour 1 from y to x (4,2)
Target 2 reached (4,2)
empty x (0,2).

// Fill x

```
if (current state .x < x) {
    State next state = {x, current state .y};
if (visited .find (nextstate) == visited .end ()){
    cout << "Fill x (" << nextstate .x << "," <<
    nextstate .y << ")" << endl;
    pourwater (x, y, target, nextstate, visited);
    }
  }
}
int main (){
      int x, y, target;
    cout << "Enter capacity of jug x: ";
  cin >> x;
  cout << "Enter capacity of jug y:";
  cin >> y;
  cout << "Enter target amount:";
 cin >> target;
state initial state = {0,0}
unordered-set <State> visited;
cout << "solution steps :" << endl;
pourWater (x, x, target, initial state, visited);
 return 0;
}
```

Result; Jug Problem (water Jug) was successfully solved and implemented in C++.