**ARTIFICIAL INTELLIGENCE LAB**

Experiment 1

**Toy Problem**

**Q1**. What is a toy problem?

**Ans.** Toy problem or a puzzle like problem is not of immediate scientific interest, yet is used as an expository device to illustrate a trait that may be shared by other, more complicated, instances of the problem, or as a way to explain a particular, more general, problem solving technique. A toy problem is useful to test and demonstrate methodologies. In the field of artificial intelligence, classical puzzles, games and problems are often used as toy problems. These include sliding-block puzzles, N-Queens problem, missionaries and cannibals problem, tick-tack-toe, chess, Hanoi tower and others.*(source-Quora)*

**Name of Toy Problem - Missionaries and Cannibals**

In this 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, that the missionaries present on the bank cannot be outnumbered by cannibals. The boat cannot cross the river by itself with no people on board.

Operators are:

• MM: 2 missionaries cross the river

• CC: 2 cannibals cross the river

• MC: 1 missionary and 1 cannibal cross the river

• M: 1 missionary crosses the river

• C: 1 cannibal crosses the river

**Q2.** How many approaches do you have for solving the toy problem which you have taken?

**Ans.** Here I'll show an arithematic approache to solve the discussed problem. We can also use DFS and BFS to solve this toy problem.

C++ code:

#include <iostream>

#include <iomanip>

using namespace std;

class game

{

public:

int counto, i;

char left[6], right[6];

int m\_num, c\_num;

bool side;

int ml\_count, cl\_count;

int mr\_count, cr\_count;

game()

{

counto = 1;

ml\_count = cl\_count = 3;

mr\_count = cr\_count = 0;

side = false;

for (i = 0; i < 3; i++)

{

left[i] = 'M';

left[i + 3] = 'C';

right[i] = ' ';

right[i + 3] = ' ';

}

}

void get()

{

start:

cout << "\nEnter no.of missionaries= ";

cin >> m\_num;

cout << "\nEnter no.of cannibals= ";

cin >> c\_num;

if (m\_num > 3 || c\_num > 3 || m\_num < 0 || c\_num < 0)

goto start;

else if ((m\_num + c\_num) > 2 || (m\_num + c\_num == 0))

goto start;

}

void displaymc()

{

cout << "\nleft side\tright side\n";

for (i = 0; i < 3; i++)

cout << left[i] << " ";

cout << "\t\t";

for (i = 0; i < 3; i++)

cout << right[i] << " ";

cout << endl;

for (i = 3; i < 6; i++)

cout << left[i] << " ";

cout << "\t\t";

for (i = 3; i < 6; i++)

cout << right[i] << " ";

cout << endl;

if (counto % 2 == 0)

{

side = true;

cout << "\nBoat on right side of river\n";

}

else

{

side = false;

cout << "\nBoat on left side of river\n";

}

}

void boat\_lr()

{

for (i = 0; i < m\_num; i++)

{

if (left[0] == 'M')

{

left[0] = ' ';

right[0] = 'M';

ml\_count -= 1;

mr\_count += 1;

}

else if (left[1] == 'M')

{

left[1] = ' ';

right[1] = 'M';

ml\_count -= 1;

mr\_count += 1;

}

else if (left[2] == 'M')

{

left[2] = ' ';

right[2] = 'M';

ml\_count -= 1;

mr\_count += 1;

}

}

for (i = 0; i < c\_num; i++)

{

if (left[3] == 'C')

{

left[3] = ' ';

right[3] = 'C';

cl\_count -= 1;

cr\_count += 1;

}

else if (left[4] == 'C')

{

left[4] = ' ';

right[4] = 'C';

cl\_count -= 1;

cr\_count += 1;

}

else if (left[5] == 'C')

{

left[5] = ' ';

right[5] = 'C';

cl\_count -= 1;

cr\_count += 1;

}

}

}

void boat\_rl()

{

for (i = 0; i < m\_num; i++)

{

if (right[0] == 'M')

{

right[0] = ' ';

left[0] = 'M';

ml\_count += 1;

mr\_count -= 1;

}

else if (right[1] == 'M')

{

right[1] = ' ';

left[1] = 'M';

ml\_count += 1;

mr\_count -= 1;

}

else if (right[2] == 'M')

{

right[2] = ' ';

left[2] = 'M';

ml\_count += 1;

mr\_count -= 1;

}

}

for (i = 0; i < c\_num; i++)

{

if (right[3] == 'C')

{

right[3] = ' ';

left[3] = 'C';

cl\_count += 1;

cr\_count -= 1;

}

else if (right[4] == 'C')

{

right[4] = ' ';

left[4] = 'C';

cl\_count += 1;

cr\_count -= 1;

}

else if (right[5] == 'C')

{

right[5] = ' ';

left[5] = 'C';

cl\_count += 1;

cr\_count -= 1;

}

}

}

};

int main()

{

game g;

while (true)

{

if (g.ml\_count < g.cl\_count && g.ml\_count > 0 || g.mr\_count < g.cr\_count && g.mr\_count > 0)

{

cout << "\n\n~~~~~~~~~~~~~~~~YOU LOST!!~~~~~~~~~~~~~~~~\n";

break;

}

else if (g.cr\_count == g.mr\_count && g.cr\_count == 3 && g.mr\_count == 3)

{

cout << "\n\n~~~~~~~~~~~~~~~~YOU WON!!~~~~~~~~~~~~~~~~\n";

break;

}

else

{

g.displaymc();

g.get();

if (g.side == false)

g.boat\_lr();

else

g.boat\_rl();

}

g.counto++;

}

return 0;

}