**Name**: Raj K Patel

**Roll No**: 20BCE218

**Course**: 2CSDE93 - Blockchain Technology

Practical No: 4

**Aim**: Practical byzantine fault tolerance in Blockchain systems, the math behind this algorithm.

Code:

#include <bits/stdc++.h> using namespace std;

enum Result

**{**

Attack, Retreat, Conflict

**};**

int main()

**{**

cout << "Enter the string: \n"; cout << "\tL => Loyal\n";

cout << "\tT => Traitor\n"; string input = "LLLL";

for (int i = 0; i < 4; i++)

**{**

cin >> input[i];

**}**

cout << "Provide an action: Attack or Retreat\n"; string s\_action;

cin >> s\_action;

Result action = Result::Conflict;

if (s\_action == "Attack")

action = Result::Attack; else if (s\_action == "Retreat")

action = Result::Retreat; else

**{**

cout << "ERROR: Incorrect input\n"; exit(1);

**}**

map<int, vector<Result>> res; for (int i = 0; i < 4; i++)

**{**

for (int j = 0; j < 4; j++)

**{**

if (i == j)

continue;

if (input[i] == 'T')

**{**

if (action == Result::Attack) res[j].push\_back(Result::Retreat);

else

res[j].push\_back(Result::Attack);

**}**

if (input[i] == 'L')

**{**

res[j].push\_back(action);

**}**

**}**

**}**

int overall\_attack = 0, overall\_retreat = 0; for (int i = 0; i < 4; i++)

**{**

Result ans = Result::Conflict; int attack = 0;

int retreat = 0;

for (auto it : res[i])

**{**

if (it == Result::Attack)

**{**

attack++; overall\_attack++;

**}**

if (it == Result::Retreat)

**{**

retreat++; overall\_retreat++;

**}**

**}**

cout << "Lt. " << i << " ";

cout << "Attack: " << attack << " Retreat: " << retreat

<< endl;

**}**

if (overall\_attack > overall\_retreat)

**{**

cout << "Overall Result: Attack\n";

**}**

else if (overall\_attack < overall\_retreat)

**{**

cout << "Overall Result: Retreat\n";

**}**

else

**{**

cout << "Overall Result: Conflict\n";

**}**

return 0;

**}**

Output:

