```
/*
Name: Rohit Saini
Erp: 1032200897
Panel: C
RollNo: PC-41 */
```

TITLE : Write a C Program to Implement Bully Election Algorithm.

AIM : To study and implement Bully Election Algorithm

OBJECTIVES :

1. To understand the working of election algorithms
2. To simulate bully algorithm

THEORY :

Elections -

Bully Algorithm -

Types of messages -

There are basically 3 types of messages :

1. An election message to initiate the election
2. A reply/response message given in response to the election message
3. A coordinator message sent to inform other processes, the id of the coordinator process.

INPUT : Process IDS

OUTPUT : Selected Coordinator Process

```
PS C:\Users\rohit\Documents\GitHub\sem_7\dc\lab5> .\bully_election.exe
Enter number of nodes: 6
42 170 335 468 501 725
725 has gone down

Who will conduct election?: 170
170 Sends message to 335. Recieved: OK
170 Sends message to 468. Recieved: OK
170 Sends message to 501. Recieved: OK

Node Conducting Election : 335
335 Sends message to 468. Recieved: OK
335 Sends message to 501. Recieved: OK
Nodes are now eligible to conduct elections: 468 501
Node Conducting Election : 468
468 Sends message to 501. Recieved: OK

Node Conducting Election : 501
New coordinator is Node ID: 501
501 sends message to 42
501 sends message to 170
501 sends message to 335
501 sends message to 468
PS C:\Users\rohit\Documents\GitHub\sem_7\dc\lab5>
```

PLATFORM : UNIX,

PROGRAMMING LANGUAGE : C Language

Code:

```cpp
#include <iostream>
#include <vector>
#include <windows.h>
#include <algorithm>
using namespace std;

class Node
{
public:
    int nodeId;
    bool isOnline;

    Node()
```

```
    {
        this->nodeId = rand() % 1000 + 1;
        this->isOnline = true;
    }

    int send(int initiatorId)
    {
        if (this->isOnline && this->nodeId > initiatorId)
        {
            cout << initiatorId << " Sends message to ";
            cout << this->nodeId << ". Recieved: OK" <<
endl;
            return 1;
        }
        return 0;
    }

    void shutDown()
    {
        Sleep(3);
        cout << this->nodeId << " has gone down" << endl;
        this->isOnline = false;
    }
};

class ElectionCoordinator
{
public:
    vector<Node *> network;
    Node *coordinatorNode;

    ElectionCoordinator(int numNodes)
    {
        for (int i = 0; i < numNodes; i++)
        {
            this->network.push_back(new Node());
```

```
        }
        sort(network.begin(), network.end(), [](Node *lhs,
Node *rhs)
             { return lhs->nodeId < rhs->nodeId; });
        coordinatorNode = network[numNodes - 1];
    }

    void printNetwork()
    {
        for (Node *node : network)
        {
            cout << node->nodeId << " ";
        }
        cout << endl;
    }
    void final_msg(int nodeId)
    {
        for (Node *node : network)
        {
            if (node->nodeId < nodeId)
                cout << nodeId << " sends message to " <<
node->nodeId << endl;
        }
    }
    vector<Node *> conductElection(int initiatorId)
    {
        vector<Node *> responses;
        for (Node *node : network)
        {
            if (node->send(initiatorId) == 1)
            {
                responses.push_back(node);
            }
        }
        return responses;
    }
```

```
    void selectCoordinator(int nodeId)
    {
        for (Node *node : network)
        {
            if (node->nodeId == nodeId)
                coordinatorNode = node;
        }
    }
};

int main()
{
    int numNodes;
    cout << "Enter number of nodes: ";
    cin >> numNodes;
    ElectionCoordinator election(numNodes);
    election.printNetwork();
    election.coordinatorNode->shutDown();
    int initiatorId;
    vector<Node *> eligibleNodes;
    cout << "\nWho will conduct election?: ";
    cin >> initiatorId;
    eligibleNodes = election.conductElection(initiatorId);

    while (eligibleNodes.size())
    {
        initiatorId = eligibleNodes[0]->nodeId;
        cout << "\nNode Conducting Election : " <<
initiatorId << endl;
        eligibleNodes =
election.conductElection(initiatorId);
        if (eligibleNodes.size() > 1)
        {
            cout << "Nodes are now eligible to conduct
elections: ";
```

```
            for (Node *node : eligibleNodes)
            {
                cout << node->nodeId << " ";
            }
        }
    }

    cout << "New coordinator is Node ID: " << initiatorId
<< endl;
    election.selectCoordinator(initiatorId);
    election.final_msg(initiatorId);
    return 0;
}
```

CONCLUSION : Thus, bully algorithm is successfully implemented.

FAQs

1. What is the time complexity (best,avg,worst)of bully algorithm
2. Why do we have to elect the coordinator process
3. How did the name of “Bully” approach come up ?

Page No.
Date

DS LAB-5

FAQ

Q1. What is the time complexity of bully algorithm.

Ans. Best → If the coordinator process is already known. O(1).

Worst → where all other processes must participate in an election. O(n)

average → O(n)

Q2. Why do we have to elect the coordinator process.

Ans.

1. Resource Management: Coordinator can distribute tasks or responsibility among the participating process i.e. schedule tasks, and prevent resource conflicts by making centralized decisions.
2. Load Balancing: can distribute tasks or responsibilities.
3. Fault Tolerance: if coordinator process fails or become unreachable, electing a new coordinator ensures the system's continued operations and prevent it from becoming unresponsive.
4. Synchronization: coordinator is necessary to ensure that processes perform tasks in synchronized and orderly manner.

Q3. How did the name of "Bully" approach come up?

Ans → The name Bully Algorithm is derived from the behaviour of the processes in the algorithm, when process suspects that coordinator process has failed or become unresponsive. it bullies its ways into coordinators position.