Name: Shivam Darekar
Prn: 202101040055

Div: C Batch: C3

Practical 4

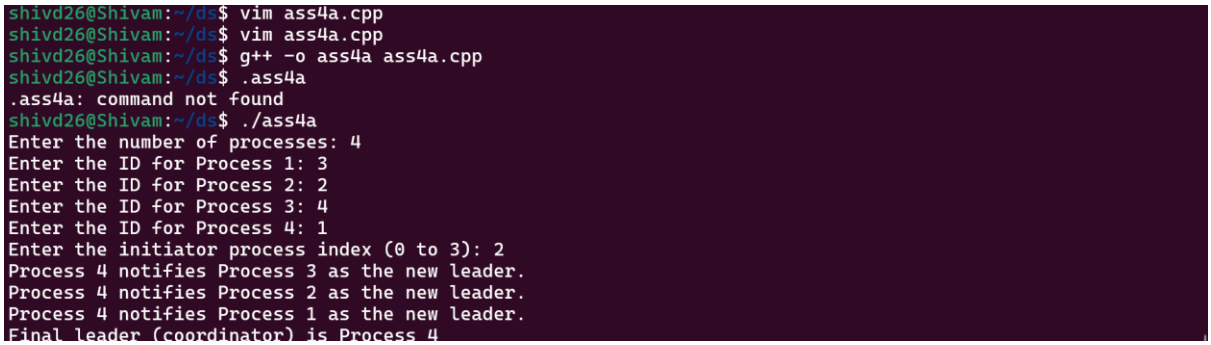
Implement various Election Algorithms for Coordinator Selection in a distributed system.

A: Bully Algorithm

```
#include <iostream>
#include <vector>
using namespace std;

void bullyAlgorithm(vector<int>& processIds, int numProcesses, int initiator) {
   int leader = -1;

   // Processes with higher IDs respond
   for (int i = 0; i < numProcesses; ++i) {
      if (processIds[i] > processIds[initiator]) {
         cout << "Process " << processIds[initiator] << " initiates election." << endl;
         cout << "Process " << processIds[i] << " responds to Process " << processIds[initiator] << endl;
         leader = processIds[i];
      }
   }

   if (leader == -1) {
      // If no higher process responds, the initiator becomes the leader
      leader = processIds[initiator];
   }

   // Announce the leader
   for (int i = 0; i < numProcesses; ++i) {
```

```
        if (processIds[i] != leader) {
            cout << "Process " << leader << " notifies Process " << processIds[i] << " as the new leader." << endl;
        }
    }

    cout << "Final leader (coordinator) is Process " << leader << endl;
}

int main() {
    int numProcesses;
    cout << "Enter the number of processes: ";
    cin >> numProcesses;

    vector<int> processIds(numProcesses);
    for (int i = 0; i < numProcesses; ++i) {
        cout << "Enter the ID for Process " << i + 1 << ": ";
        cin >> processIds[i];
    }

    int initiator;
    cout << "Enter the initiator process index (0 to " << numProcesses - 1 << "): ";
    cin >> initiator;

    bullyAlgorithm(processIds, numProcesses, initiator);

    return 0;
}
```

Output:

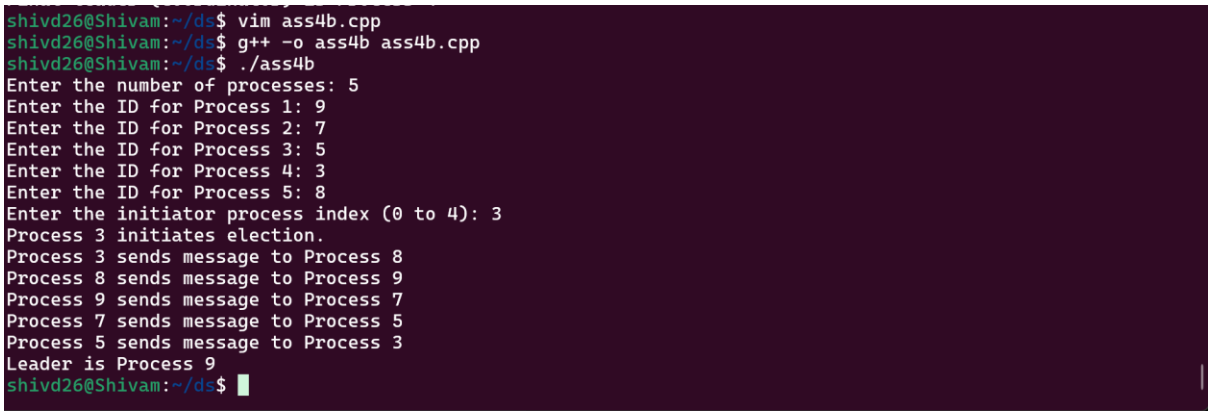


B: Ring Algorithm

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

void ringAlgorithm(vector<int>& processIds, int numProcesses, int initiator) {
    int leader = processIds[initiator];
    int current = initiator;

    cout << "Process " << processIds[initiator] << " initiates election." << endl;

    do {
        int next = (current + 1) % numProcesses;
        cout << "Process " << processIds[current] << " sends message to Process " << processIds[next] << endl;

        // Keep the highest ID as leader candidate
        leader = max(leader, processIds[next]);
        current = next;

    } while (current != initiator);
```

```
    // Announce the leader
    cout << "Leader is Process " << leader << endl;
}

int main() {
    int numProcesses;
    cout << "Enter the number of processes: ";
    cin >> numProcesses;

    vector<int> processIds(numProcesses);
    for (int i = 0; i < numProcesses; ++i) {
        cout << "Enter the ID for Process " << i + 1 << ": ";
        cin >> processIds[i];
    }

    int initiator;
    cout << "Enter the initiator process index (0 to " << numProcesses - 1 << "): ";
    cin >> initiator;

    ringAlgorithm(processIds, numProcesses, initiator);

    return 0;
}
```

OUTPUT:


shivd26@Shivam:~/ds$ vim ass4b.cpp
shivd26@Shivam:~/ds$ g++ -o ass4b ass4b.cpp
shivd26@Shivam:~/ds$ ./ass4b
Enter the number of processes: 5
Enter the ID for Process 1: 9
Enter the ID for Process 2: 7
Enter the ID for Process 3: 5
Enter the ID for Process 4: 3
Enter the ID for Process 5: 8
Enter the initiator process index (0 to 4): 3
Process 3 initiates election.
Process 3 sends message to Process 8
Process 8 sends message to Process 9
Process 9 sends message to Process 7
Process 7 sends message to Process 5
Process 5 sends message to Process 3
Leader is Process 9
shivd26@Shivam:~/ds$