CPSC-474 – DOCUMENTATION
RING LEADER ELECTION USES ODD EVEN FORMULA

Group Members:

Balwinder Singh Hayer
Saytu Singh

How to Execute program:

```
mpicc -o ring ring.c
mpirun -np 4 ./ring
```

[When we try to execute program, make sure program is in current directory]

PSEUDO-CODE:

Integer Function [calculate_random_value] (int world_rank, int random)

```
 1.Generate a random value
 2. Compare the random value for < 100 and > 0
 if (random < 0)
    random = abs(random);
 if (random <= 10)
    random = random + 10;
 if (random > 100)
  random = random % 100;

 random = 1000 + (world_rank * 100) + random ;

int compute_value(int frandom){
Test if frandom is even or odd
 if (frandom % 2 == 0)
 {
  frandom = frandom;
  printf("Even\n");
  return (0);
 }
 else
 {
  printf("odd\n");
  return (1);
 }
}

void even_election(int world_rank, int world_size){
 //Even Leader Election Algorithm
 int random, frandom, flag, received_value, local_value;
```

```
  srand(time(NULL));
  int even_value_counter = 0;
  int even_leader = 0;

  if (world_rank == 0) {
     random = (rand() % 7);
     frandom = calculate_random_value( world_rank, random);
     printf("Intial random value %d\n",frandom );

     flag = compute_value(frandom);
     if (flag == 1)
       frandom = frandom + 1;


     printf("Random value generated at Process %d is %d\n",world_rank,frandom );

     MPI_Send(&frandom, 1, MPI_INT, (world_rank + 1) % world_size, 0,
              MPI_COMM_WORLD);
     even_value_counter = even_value_counter + 1;

     MPI_Recv(&frandom, 1, MPI_INT, world_size - 1, 0, MPI_COMM_WORLD,
              MPI_STATUS_IGNORE);
     //printf("Leader %d\n", even_leader);

     printf("Process %d received token %d from process %d\n", world_rank, frandom,
          world_size - 1);
  }

  if (world_rank != 0) {

     MPI_Recv(&frandom, 1, MPI_INT, world_rank - 1, 0, MPI_COMM_WORLD,
              MPI_STATUS_IGNORE);

     received_value = frandom;

     random = (rand() % world_rank);
     local_value = calculate_random_value(world_rank, random);
     printf("Local value computed is %d\n",local_value );
     flag = compute_value(local_value);



     if (flag == 1)
     {
       MPI_Send(&received_value, 1, MPI_INT, (world_rank + 1) % world_size, 0,
MPI_COMM_WORLD);
       printf("At Process:%d received_value is %d and flag is %d\n", world_rank, received_value, flag);
     }
     else if (flag == 0)
```

```
    {
            if (received_value > local_value)
               {
                  MPI_Send(&received_value, 1, MPI_INT, (world_rank + 1) % world_size, 0,
MPI_COMM_WORLD);
                  even_leader = world_rank-1;
                }

                else if (local_value > received_value)
                  {
                    MPI_Send(&local_value, 1, MPI_INT, (world_rank + 1) % world_size, 0,
                          MPI_COMM_WORLD);
                    even_leader = world_rank;
                  }
                    printf("At Process:%d received_value is %d and flag is %d\n", world_rank,
received_value, flag);
    }


    printf("Process %d received token %d from process %d\n", world_rank, local_value,
         world_rank - 1);
 }

   printf("***************Even Leader Election*****************\n");

}


void odd_election(int world_rank, int world_size){
    //Odd Leader Election Algorithm
 int random, frandom,flag, received_value, local_value;
 srand(time(NULL));
 int even_value_counter = 0;
 int even_leader = 0;

 if (world_rank == 0) {
    random = (rand() % 7);
    frandom = calculate_random_value( world_rank, random);
    printf("Intial random value %d\n",frandom );

    flag = compute_value(frandom);
    if (flag == 0)
      frandom = frandom + 1;


    printf("Random value generated at Process %d is %d\n",world_rank,frandom );

    MPI_Send(&frandom, 1, MPI_INT, (world_rank + 1) % world_size, 0,
          MPI_COMM_WORLD);
```

```
    even_value_counter = even_value_counter + 1;

    MPI_Recv(&frandom, 1, MPI_INT, world_size - 1, 0, MPI_COMM_WORLD,
            MPI_STATUS_IGNORE);
    //printf("Leader %d\n", even_leader);

    printf("Process %d received token %d from process %d\n", world_rank, frandom,
          world_size - 1);
}

else if (world_rank != 0) {

    MPI_Recv(&frandom, 1, MPI_INT, world_rank - 1, 0, MPI_COMM_WORLD,
            MPI_STATUS_IGNORE);

    received_value = frandom;

    random = (rand() % world_rank);
    local_value = calculate_random_value( world_rank, random);
    printf("Local value computed is %d\n",local_value );
    flag = compute_value(local_value);

    if (flag == 0)
    {
      MPI_Send(&received_value, 1, MPI_INT, (world_rank + 1) % world_size, 0,
                MPI_COMM_WORLD);
      printf("At Process:%d received_value is %d and flag is %d\n", world_rank, received_value, flag);
    }
    else if (flag == 1)
    {
        if (received_value > local_value)
        {
            MPI_Send(&received_value, 1, MPI_INT, (world_rank + 1) % world_size, 0,
                  MPI_COMM_WORLD);
            even_leader = world_rank-1;
        }

        else if (local_value > received_value)
        {
            MPI_Send(&local_value, 1, MPI_INT, (world_rank + 1) % world_size, 0,
                              MPI_COMM_WORLD);
            even_leader = world_rank;
        }
      printf("At Process:%d received_value is %d and flag is %d\n", world_rank, received_value, flag);
    }

    printf("Process %d received token %d from process %d\n", world_rank, local_value,
          world_rank - 1);
```

```
  }

    printf("***************Odd Leader Election*****************\n");

}


int main(int argc, char** argv){

  MPI_Init(NULL, NULL);

  int world_rank, flag, received_value, local_value;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);
  int random, frandom;
  srand(time(NULL));
  int even_value_counter = 0;
  int even_leader = 0;

  // Program terminates within 60 minutes if N = 10
  if (world_size == 10) {
       exit(1);
  } else if (world_size >= 6 || world_size <= 20) {

     even_election(world_rank, world_size);

     odd_election(world_rank, world_size);
  } else {
     printf("N or Number of processes must be between 6 and 20");
  }

  MPI_Finalize();

  return 0;
}
```

OUTPUT:

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/oddEven$ mpicc -o ring ring.c
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/oddEven$ mpirun -np 4 ./ring
Intial random value 1016
Even
Random value generated at Process 0 is 1016
Local value computed is 1110
Even
At Process:1 received_value is 1016 and flag is 0
```

Process 1 received token 1110 from process 0
**********Even Leader Election**********
Local value computed is 1210
Even
At Process:2 received_value is 1110 and flag is 0
Local value computed is 1311
odd
At Process:3 received_value is 1210 and flag is 1
Process 0 received token 1210 from process 3
**********Even Leader Election**********
Initial random value 1016
Even
Random value generated at Process 0 is 1017
Local value computed is 1110
Even
At Process:1 received_value is 1017 and flag is 0
Process 1 received token 1110 from process 0
**********Odd Leader Election**********
Process 3 received token 1311 from process 2
**********Even Leader Election**********
Process 2 received token 1210 from process 1
**********Even Leader Election**********
Local value computed is 1210
Even
At Process:2 received_value is 1017 and flag is 0
Process 2 received token 1210 from process 1
**********Odd Leader Election**********
Local value computed is 1311
odd
At Process:3 received_value is 1017 and flag is 1
Process 0 received token 1311 from process 3
**********Odd Leader Election**********
Process 3 received token 1311 from process 2
**********Odd Leader Election**********