# Project 2 Submission

## Names and CSUF-supplied email addresses


Pooja Honneshwari Ravi pooja.ravi@csu.fullerton.edu
Ameena Khan ameena07@csu.fullerton.edu


## Summary of Report Document


This report document, for project 2, contains the pseudocode of the algorithm and three screenshots. One screenshot is of the group members and the rest are snapshots of the code executing. It also contains a brief description of how to run the code


## A full-screen screenshot with group member names

# 474-Project2

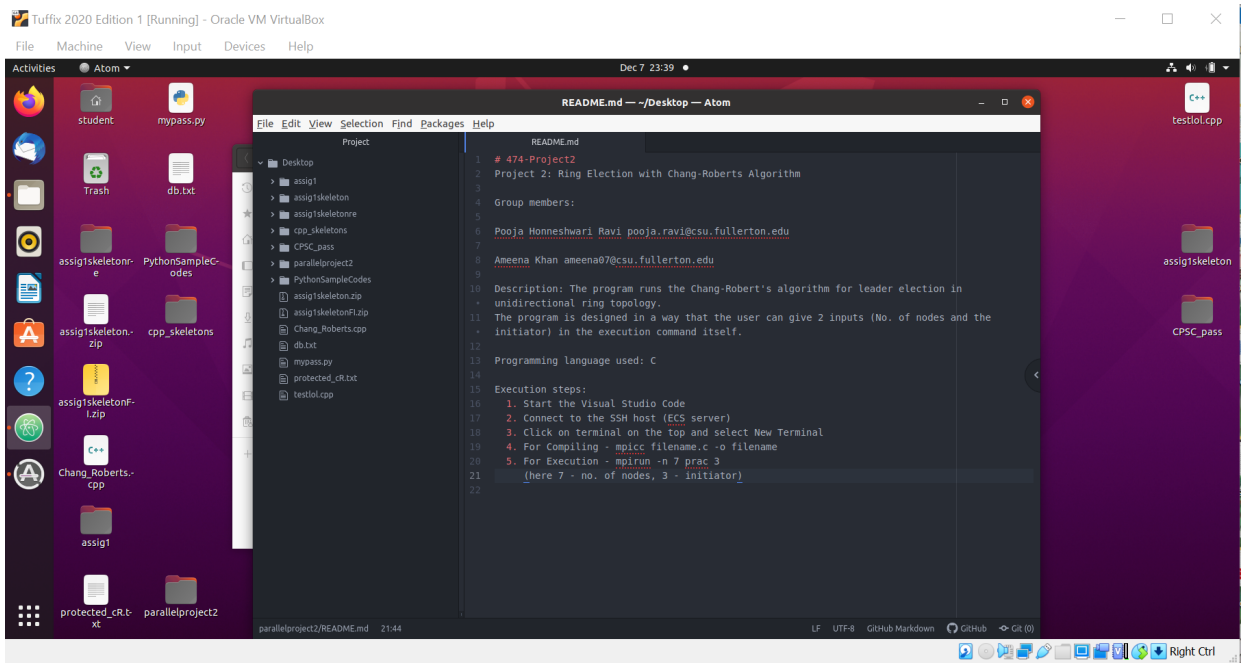
Project 2: Ring Election with Chang-Roberts Algorithm

Group members:

Pooja Honneshwari Ravi pooja.ravi@csu.fullerton.edu

Ameena Khan ameena07@csu.fullerton.edu

Description: The program runs the Chang-Robert's algorithm for leader election in unidirectional ring topology. The program is designed in a way that the user can give 2 inputs (No. of nodes and the initiator) in the execution command itself.



**Pseudocode**

```
//function Chang Roberts Algo

//from terminal user enters number of nodes and which node should be initiator

//let passive state be 0
//let active state be 1

//set up random number generator

//user choose which processor will be initiator (in terminal)
//this will start the election process

//while loop
        //election finished
        if
                //election still in process
                Else{
                        //if (node == to its own value) → (msgPkt[1] == UID)
                                //If you end up here, leader has been found
                                //so set up everything so that the next time the while loop
                                //runs, it will trigger the FIRST if statement, which
                                //can then send out election message and then exit while loop
                                //leader determined = true
                                //msgPkt[2] = UID
                                //msgPkt[0] = ELECTED_MSG

                        //else if(node less than value) →(msgPkt[1] > UID)
                                //become passive
                                //if won't do anything else (DO NOT pass value)
                        //else if(node greater than value) →(msgPkt[1] < UID)
                                //active process
                                //msgPkt[1] = UID

                        //print state


                        }
// Send message to the next neighbor process

//once done with the previous while loop, leader has been found
//now pass that msg to to all remaining nodes and put them in passive state

//while electedMsgPassing is true

        //print status

        //if not the leader —> (msgPkt[0] == ELECTED_MSG && UID != leaderUID)
                //pass on msg of who is leader
                //make passive (could possibly be passive already)
```

```
        //else if (msgPkt[0] == ELECTED_MSG && UID == leaderUID)
                //electdMsgPassing = false

//leader has been found
//if leader found
        //print the leader rank and unique id
        //printf("\n******\tLeader!!! Rank: %d UID: %d\t******\n", rank, UID);

//error checking
//a processes rank should be between 0 and size-1

//function to check if valid
        //if rank >= size
                //print error statement
                Return false

        //if we manage to get here, then everything is ok
        //return true

//main function
        //initial setup

        //Three processes are a minimum to simulate the chang roberts algo
        //check if we have at LEAST 3 processes
        //if commSize < 3
                //Print error statement….need at least 3 processes!


        //from the user input (from terminal) get which node will initiate
        //if (procRank == ZERO_RANK_PROCESS)
                //print “election will be initiated by [rank]”

        //make function calls

        //return 0
```

**A brief description on how to run the code.**

1. Start Visual Studio Code
2. Connect to the SSH host (ECS Server)
3. Click on terminal on the top and select new terminal
4. For compiling - mpicc filename.c -o filename
5. For execution - mpirun -n 7 prac 3 (here 7 - no. of nodes, 3 - initiator)

**Snapshots of code executed at least once**. Alternatively one can create a file with the output of the program for an input value and submit it together with the program. Note, the output can be redirected to a file (for easy printing). For example, the following command line will create an output file in Linux-based operating system called a1out.txt by re-directing the output from the screen (display) to the file a1out.txt:

K:\cs474> a.out > a1out.txt