Communication and Information Engineering
Parallel and Distributed Computing (CIE564)
Instructors: **Dr. Anas Youssef**
TAs: **Hanan Khaled and Reem Abdel-salam**

Phase 1 Project Description

Project Objective

Managing the election process to choose a new president

Project Statement

The results of any elections may take weeks to be announced. But we want to quickly announce which candidate will win. So given the voters' preferences lists, you need to write a parallel program to announce which candidate will win and in which round.

Project Description

When it is time to vote for a new president and as a voter you are really excited about that. You know that the final results may take weeks to be announced, while you can't really wait to see the results. Suppose the preferences list is available for every voter. Each voter sorted out all candidates starting by his most preferred candidate and ending with his least preferred one. When voting, a voter votes for the candidate who comes first in his preferences list. For example, if there are 5 candidates (numbered 1 to 5), and the preferences list for one voter is [3, 2, 5, 1, 4] then voter will give the highest vote for candidate 3 and the lowest vote for candidate 4.

The rules for the election process are as follows:

1. There are C candidates (numbered from 1 to C), and V voters.
2. The election process consists of up to 2 rounds. All candidates compete in the first round. If a candidate receives more than 50% of the votes, he wins, otherwise another round takes place, in which only the top 2 candidates compete for the presidency, the candidate who receives more votes than his opponent wins and becomes the new president.
3. The voters' preferences are the same in both rounds so if the preference list [1 2 3 4 5] in the first round and the second round become between candidate 1 and 2 so the preferences is the same [1 2].

Given the preferences lists, you need to write a parallel program to announce which candidate will win and in which round.

For example: If the input is
3 5 // number of candidates & number of voters
1 2 3 // voter 1 preference list
1 2 3 // voter 2 preference list
2 1 3 // voter 3 preference list
2 3 1 // voter 4 preference list
3 2 1 // voter 5 preference list
Then the output will be 2 2 // candidate 2 wins in round 2

Explanation: You should print the output something like this:
Candidate [1] got 2/5 which is 40%
Candidate [2] got 2/5 which is 40%
Candidate [3] got 1/5 which is 20%
So second round will take place between candidates 1 and 2 with same preferences
1 2 // voter 1 preference list
1 2 // voter 2 preference list
2 1 // voter 3 preference list
2 1 // voter 4 preference list
2 1 // voter 5 preference list
Candidate [1] got 2/5 which is 40%
Candidate [2] got 3/5 which is 60% so candidate 2 wins in round 2

And if the input is
2 3 // number of candidates & number of voters
2 1 // voter 1 preference list
1 2 // voter 2 preference list
2 1 // voter 3 preference list

Then the output will be 2 1 // candidate 2 wins in round 1

The code should generate **a data file** that contains the voters' randomly generated preferences. The format of the file must be as follows: number of candidates in the first line, number of voters in the second line, and voters' preferences equal to number of voters. The format of the file is shown in **Figure 1**.

Run Steps that should be followed:

When the program run you should prompt the user to:

1. Generate a data file like the one showed in Figure 1 to generate the voter's preferences
2. Calculate the result and print the winner and in which round using the file generated from step 1 as input.

Note; when running your program, the file must not be loaded by one process but every running process should loads its part from the file.

The Expected Output:

Print to the console the steps happening in every process and print which candidate will win the elections and in which round. If there are 2 rounds, print this information and show the percentage obtained by every candidate per each round.

Each process will calculate how many each candidate was mentioned as a first preference and will add that to a master process (process 0) that will store all preferences

then normalize if any is more than 0.5 then he wins directly

if not then we initiate second round between the most prefered 2 candidates

filter candidates except the most 2 prefered (from the origional file) and a great point to speed things up
build upon the last preference count all we are gonna do is to just count lines that don't have the most 2 prefered candidates as their first choice

(to build upon that each round will write to a file round 1

win / round2
candidate_k number
candidate_m number

init code finalize
if(round2) init code finalize

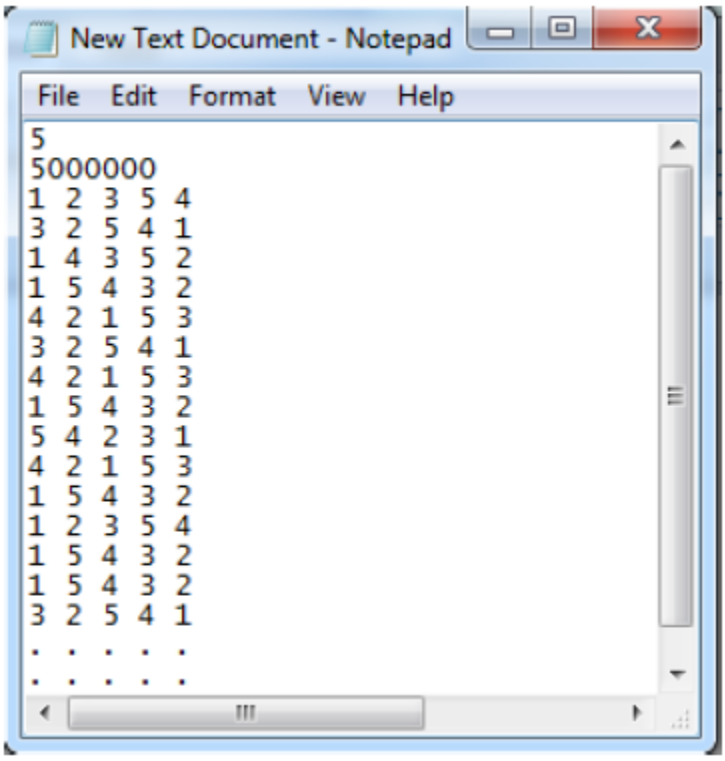

Figure 1

Deliverables and Submission Instructions:

1. The project team is at most three members.
2. If any team consists of four members, **Bonus requirement is a mandatory.**
3. Submit single zip file containing C source Code must be written using MPI library.
4. The zip file should also contain a PDF document that contains your team members' names and ids and that describes the implemented parallel scenario and the steps followed in the source code.

Marking Criteria:

The phase 1 project's grade is out of 10% distributed as follows:

Phase 1 – Midnight- Sunday Dec 13, 2020	
1- Handling file read = 0.5 %	4- Handling second round = 3.5 %
2- Handling file generation = 0.5 %	5- Print output = 0.5%
3- Handling first round = 4%	6- Submitting PDF document = 1 %

Additional Requirement as a Bonus:

File generation as shown in figure 1 should be implemented in a **parallel** manner.