Problems on Process Co-ordination

The goal of this project is to solve some problems involving process co-ordination, in C using pthreads and semaphores. Grading of this project will be based on two criteria, correctness and presentation of code, and report that you will submit.

For each problem, you have to do the following in your report:

1. Identify the correctness constraints for each problem
2. Specify the conditions where wait should happen and justify your reasons for them.
3. Also specify, when your algorithm can lead to starvation, deadlock or race condition etc, and if not why
4. Implement the solution using semaphores only.

Problem 1 Editors-Reporters Problem

It is the world of 2021, IIIT has now #n refined departments in the field of C.S and IIIT takes a lot of pride, in its local run magazine, PING. But since there are so many departments, everybody wants a piece, in PING. For that purpose, our parliament had appointed editors from each department. Since it is a competitive world, each editor wants to have best entries, so each editor takes all reports that were sent to ping@students.iiit.ac.in. But soon, different editors were publishing same material, and which degraded the quality of PING

Now at this time, the Institute remembered that in year of 2012, the guys who took OS course, did a marvelous job of solving this problem, so lets switch to present and your task is to solve the following problem so that you can save chaos of future.

The following constraints are there:

- There are exactly #n editors and #m no. of reports/article.
- Each editor will randomly pick an article, and will either accept it or reject it. For example, generate a random number, if number is even accept or number is odd reject. You can even have your own probability distribution. • If an editor rejects an article, he will never accept it again
- Each editor requires exactly one second to read an article, while reading no other editor can read the same article.
- If an editor is reading an article, other editors should not wait on the same article.
- If an editor accepts an article, no other editor can read it and that implies removal of that article from list of articles.
- Program ends when there is no article that can be accepted by any editor. • Not necessarily, all articles are published.

Input:
./a.out n m

Output:
Write n lines, each ith line corresponding to ith editor. Each line starts with no. of articles accepted, followed by indices of articles he accepted (indexing from 1 to m).

Problem 2 Dish-Washing Problem

In a particular mess of IIIT, we have a dish washing junction, but there are only n taps, n-1 scrotch-brites with soap-dishes (Consider the arrangement at NBH). Now a guy has a plate, spoon and glass with him, that he wants to wash, under the following constraints:

- There are two actions, either he can rub a utensil with scrotch brite or he can wash a utensil with tap water. Ensure that a utensil is washed with water only after it has been rubbed with scrotch brite.
- A person should acquire a free tap first and will continue to occupy the same spot till he has washed all the utensils. If a person is in wait for tap, then he continues to wait, there is no preference over user.
- Once a person has found a tap, he will start looking for a free scrotch brite, only next to his tap. He must co-ordinate with his neighbors to wash three utensils such that he finishes as soon as he can, via synchronizing over scrotch brite.
- Arrangement is of the form: TSTSTST (where T->Tap, S->Scrotch Brite, n == 4).
- Time requirements is as follows
-

Plate Glass Spoon	With scrotch brite	With tap/water
	4	5
	3	3
	2	1

- It is possible that while a person is using tap, some neighbour can use the adjacent scrotch brite, but vice versa is not possible.

Input:
./a.out n m, where n is no. of taps and m is no. of students

Output:
Assume all of m students are in queue at t=0, and all the taps are free, output m lines, each stating at what time ith student washed all his utensils

Problem 3 Concurrent Merge Sort

- Given a number **n** and n numbers, sort the numbers using Merge Sort.
- Recursively make two child **processes**, one for the left half, one for the right half. If the number of elements in the array for a process is less than 5, perform a selection sort.
- The parent of the two children then merges the result and returns back to the parent and so on.
- Compare the performance of the merge sort with a normal merge sort implementation and make a **report**.
- You must use the **shmget, shmat** functions as taught in the tutorial.

Bonus:

- Use threads in place of processes for Concurrent Merge Sort. Add the performance comparison to the above report.

Precautions:

- Do not Copy, otherwise, you'll get a F
- You have to consider details of the algorithm yourself
- You are also entitled to assume certain viewpoints about your problem, which you have to write in your report
- Your report should be in PDF format. Be concise and do not explain unnecessarily.

Guidelines:

➔ Make sure you write a readme which briefly describes the implementation for Problems 1 and 2 (design idea) and tells how to run the code for each of the questions.