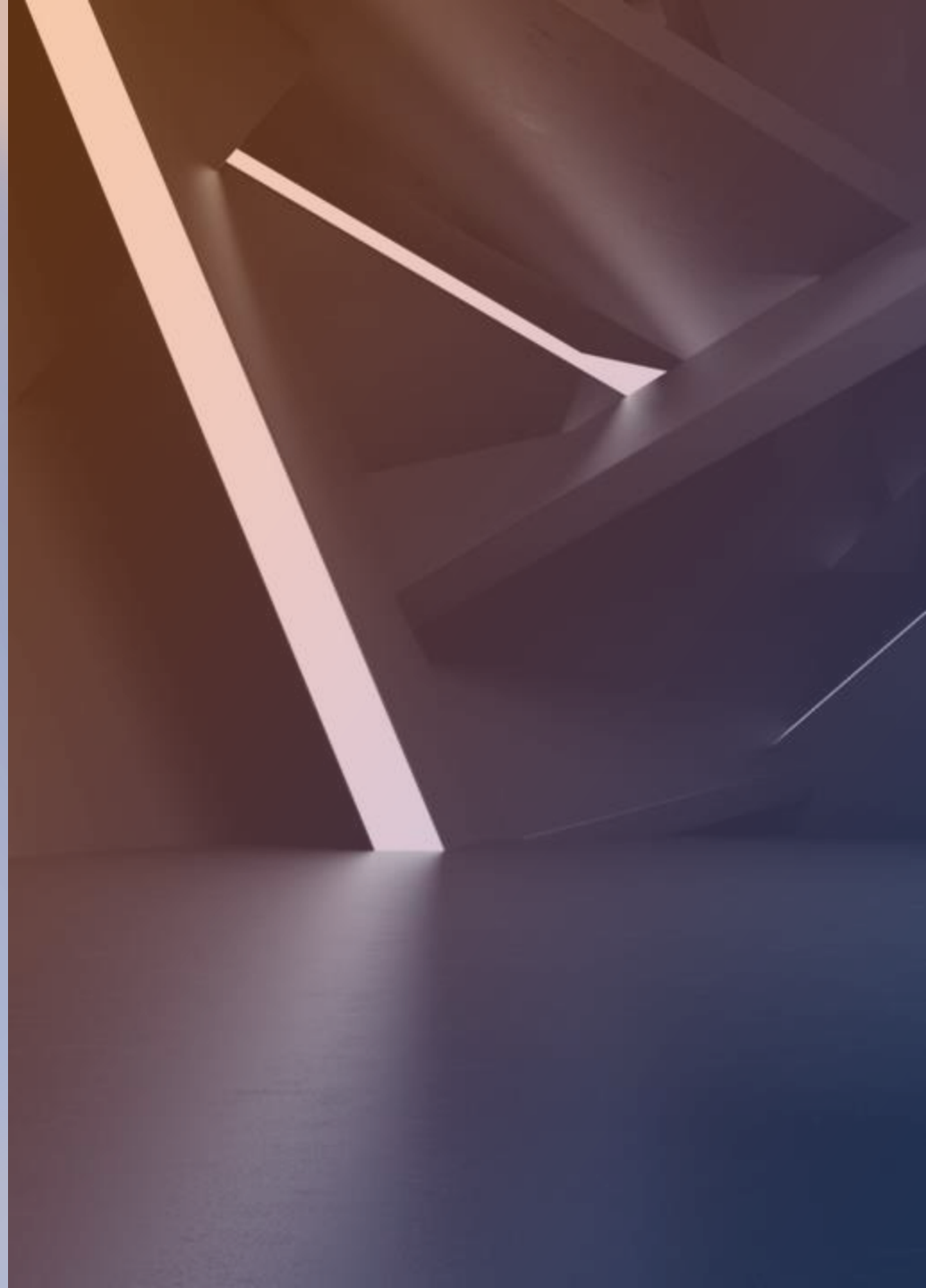
CS19003: Programming and Data Structure Lab

Lab Test 3

Sandip Chakraborty



Read Carefully

- The test will start at 4:00 pm and will end at 5:00 pm. You'll get time **till 5:15 pm** to upload your solution in the Moodle.
- **You do not need to login to Teams during the Lab test**. Just download the problem statement, solve it, and upload the solution to Moodle.
- The deadline is hard. Submission through Moodle will be considered only. No email submission will be considered at any circumstances.

Read Carefully

- Check your submission after uploading it to Moodle. We'll not attain any queries or reconsider if you upload a wrong file by mistake. Be careful with your submission.
- Read all instructions in the question carefully. No queries will be entertained during the lab exam. If you have any assumption, write it down in your solution file within comments.

Submission Requirements

- On top of your code, enter your name and roll numbers within comments. **This is essential for your submission to be considered.**

```
/**********************************
 * Name: Your Name
 * Roll: Your Roll
 **********************************/
```

Zero Tolerance to Plagiarism

- **You are supposed to write the entire code yourself**
- Do not copy the code / a block of code from any other sources (Internet, or from your friends)
- We'll take a zero-tolerance policy against plagiarism – if a significant portion of the code is similar between two students, both will be given zero.
- We'll not entertain any alibi like "both of us have copied from the same Internet source", "I had shared my Moodle password to my fried", etc.

Submission Instructions

- Once you are done with a code, save the code in your local machine as LT3.c
- Upload LT3.c in Moodle course page, corresponding to "Lab Test 3" -- https://moodlecse.iitkgp.ac.in/
- **No late submission or email submission is allowed.**

Problem

Consider that there is a party going on. There are some people in the party, who are popular in the society; therefore, all other peoples in the party know them directly (have some relation) or indirectly (do not have a direct relation but knows the person). However, it is likely that the popular person does not know everyone in the party.

Assume that there are `N` persons in the party, and this known to relationship is represented with a `NxN` matrix `Known[][]`. The values of the matrix are populated as follows.

- `Known[i][j]` = 1 if the person `i` directly knows person `j`
- `Known[i][j]` = 0 if the person `i` knows person `j` indirectly
- `Known[i][j]` = -1 if the person `i` doesn't not know person `j`

Important Notes

- There are few use cases of the problem given next. Check the use cases given under the Marks distribution.
- Even if you are not been able to solve all the user cases, try to solve some correctly to get partial marks.
- Note that your code should produce correct output corresponding to the use cases, to enable you to get the partial marks.

Problem – Cont.

- Your task will be to write a C function `int* popular (int ** Known, int N)` that takes the `Known[][]` matrix as the input and returns an array of popular persons in the party, where the first element of the array denotes the number of popular persons in the party, and the remaining elements indicate the IDs of those persons. For example, if the return array is `{3,6,9,10}`, then it indicates that there are 3 popular persons in the party, and the ID of those persons are 6, 9 and 10.
- Take the input `N` and `Known[][]` from the main function (input will be given through the terminal), call the `popular()` function, and then print the total number of popular persons and their IDs.

Example - 1

```
Enter the number of persons: 3
Enter the matrix Known[][] :
 1  0 -1
-1  1 -1
-1  1  1
Number of popular persons: 1
ID of the popular persons: 1
```

Here 0 knows 1 indirectly, and 2 knows 1 directly. However, 1 does not know others. So, 1 is a popular person. Note that every person knows himself or herself directly, so the diagonal entries of the matrix is always 1.

Example - 2

```
Enter the number of persons: 3
Enter the matrix Known[][] :
 1  0 -1
 1  1  0
-1  1  1
Number of popular persons: 0
ID of the popular persons:
```

Here 0 knows 1 indirectly, and 2 knows 1 directly. However, 1 knows 0 directly and 2 indirectly. So, 1 is not a popular person. There is no popular person in this case.

Example - 3

```
Enter the number of persons: 3
Enter the matrix Known[][] :
 1  0 -1
 1  1 -1
 1  1  1
Number of popular persons: 2
ID of the popular persons: 0   1
```

Here both 1 and 2 know 0 directly. Further, 0 knows 1 indirectly, and 2 knows 1 directly. However, both 0 and 1 do not know 2. So, both 0 and 1 are popular persons.

Important Notes

- If there are `N` persons in the party, the IDs of the persons are `0` to `(N-1)`
- A person `j` is popular if the following two conditions are satisfied.
  - Everyone knows `j` directly or indirectly
  - `j` does not know all the persons (either directly or indirectly) in the party
- The memory for the matrix `Known[][]` needs to be allocated dynamically depending on the value of `N`.

Marks Distribution

- Dynamically allocate the memory: 1 Mark
- Define and call the function properly: 1 Mark
- Return the value properly (organization of the array – first element is the count, and remaining as IDs) from the function and capture it from `main()`: 1 Mark
- 2 Marks for correct output with each of the following use cases (no marks will be given if the code does not compile successfully or throws a runtime error):
  - Everyone knows each other directly or indirectly – no popular person
  - The popular person is known to others directly only
  - The popular person is known to others indirectly only
  - There are more than one popular persons, both/all known to others indirectly only
  - There are more than one popular persons, known to others directly or indirectly
  - There is no popular person – everyone knows some of the others