B CSE221: Algorithms (Lab)

Semester: **Summer 2023** Examination: **Lab Final**

Duration: **75 minutes** Full marks: **20**

Part 1

You're in charge of optimizing the route planning for your team of elite courier agents who need to navigate through a series of delivery routes of ***N*** junctions. Each delivery route consists of several **stops that need to be completed in a specific order**. However, for collecting the packages and delivering, some stops have requirements – **certain stops that must be visited before proceeding to others**. Your task is to develop an algorithm that helps organize the delivery route in a way that respects the prerequisites and allows for efficient completion of the deliveries.

Part 2

To simplify the whole system, you decide that the **destination (the last stop) should always be represented with the largest number**. Meaning the last place in the sequence will always be the place with the largest number, **regardless of relations**. Make adequate modifications to the solution to Part 1 to accommodate this change.

Input

- You should use the **same input file for both parts**.
- The first input line has two integers ***N*** `(1 <= N <= 1000)` and ***M*** `(1 <= M <= N^2)` - the total number of stops and requirements. The stops are numbered `1, 2, 3, ... , N`.
- The next ***M*** lines describe each of the requirements. Each line has two integers ***A***, ***B*** `(1 <= A, B <= N)` - location ***A*** has to be visited before location ***B***.

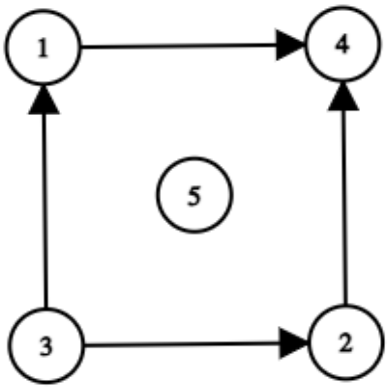
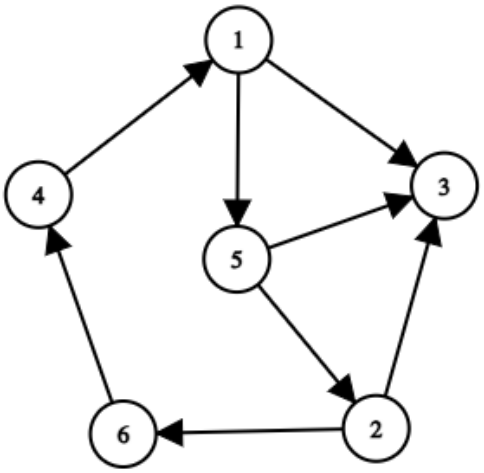
Output

- Output an adjacency list for the given street-network.
- For each part:
  - Output an order in which the route can be completed efficiently. Please note, there could be multiple correct sequences. You can print any valid order that includes all the locations.
  - If a sequence cannot be made, output **`"Cannot make route"`** instead.
- You should output to the **same file for both parts**.

Marks breakdown

- Input from file `------------>` **`2`**
- Display adjacency list `------->` **`4`**
- Implement Part 1 `----------->` **`7`**
- Implement Part 2 `----------->` **`5`**
- Output to file `-------------->` **`2`**
- **Total** `---------------------> 20`

(continued on next page)

Sample Input 1:	**Sample Output 1:**	
5 4 1 4 2 4 3 1 3 2	Adjacency List: 1 : 4 2 : 4 3 : 1 2 4 : 5 : Before adding rule: 3 5 1 2 4 After adding rule: 3 1 2 4 5	
Sample Input 2:	**Sample Output 2:**	
6 8 1 3 1 5 2 3 2 6 4 1 5 2 5 3 6 4	Adjacency List: 1 : 3 5 2 : 3 6 3 : 4 : 1 5 : 2 3 6 : 4 Before adding rule: Cannot make route After adding rule: Cannot make route	

End