CS29003 Algorithms Laboratory
Assignment 6: How do Graphs Appear from Nowhere?

General instruction to be followed strictly

1. Do not use any global variable unless you are explicitly instructed so.
2. Do not use Standard Template Library (STL) of C++.
3. Use proper indentation in your code and comment.
4. **Name your file as `<roll_no>_<assignment_no>`. For example, if your roll number is 14CS10001 and you are submitting assignment 3, then name your file as `14CS10001_3.c` or `14CS10001_3.cpp` as applicable.**
5. **Write your name, roll number, and assignment number at the beginning of your program.**
6. Make your program as efficient as possible. Follow best practices of programming.
7. Submit your program on Moodle before deadline. Submissions by email or any other means will NOT be considered for evaluation.

Suppose there are n tasks — number them 0 to $n-1$. There exist inter-dependencies among the tasks. For example, some task i may be done only after some other tasks, say j, k, ℓ are finished. User writes a dependency as two vertices j i meaning j must finish before i; refer to sample output below. User writes -1 when he/she has finished writing all dependencies. User ensures that each dependency is input exactly once. Create a suitable directed graph with these data. Use adjacency list representation.

For every job i, print all the jobs that must be finished before i. No need to print in sorted order.

A *valid plan* for performing these n jobs is a permutation of 0 to $n-1$ such that all tasks which must be completed before task i, appear to the left of i. Clearly, if there exists a sequence of jobs, say $i_1, \ldots, i_\ell$ such that i_t must be finished before i_{t+1} for every $t \in \{1, \ldots, \ell-1\}$ and i_ℓ must be finished before i_1, then there does not exist any valid plan. We call $i_1, \ldots, i_\ell$ a *cycle*. Take the number of jobs and all inter-dependencies as input. Output a valid plan if it exists; otherwise output that there exists a cycle. There is no need to output a cycle if it exists.

Implement the following algorithm for the above task. Run DFS on the graph constructed. If there exists a back edge, then output that there exists a cycle. Otherwise output the vertices (jobs) in the descending order of their finishing-timestamps/post-visit-labels — you can do this during the DFS traversal itself without requiring to spend $\mathcal{O}(n \log n)$ time for sorting. The run time of this algorithm is $\mathcal{O}(|V|+|E|)$ which is $\mathcal{O}(\text{number of jobs} + \text{number of inter-dependencies})$ in this case.

You are allowed to use two global variables of data type int.

Submit a single .c or .cpp file. Your code should get compiled properly by gcc or g++ compiler.

Sample Output

```
palash@palash-ThinkPad-X1-Yoga-3rd:~$ ./a.out
Write n: 7
Write all dependencies: 1 0 1 2 1 3 1 4 0 2 0 6 2 3 4 6 5 6 4 6 -1
0: 1
1:
2: 0 1
3: 1 2
4: 1 5
5:
6: 0 4
A valid plan: 1 0 2 3 5 4 6
palash@palash-ThinkPad-X1-Yoga-3rd:~$ ./a.out
Write n: 4
Write all dependencies: 0 3 1 0 2 1 3 2 3 1 -1
0: 1
1: 2 3
2: 3
3: 0
There exists a cycle
```

Policy on Plagiarism

Academic integrity is expected from all the students. You should work on the assignment/exam consulting only the material we share with you. You are required to properly mention/cite anything else you look at. Any student submitting plagiarized code will be penalized heavily. Repeated violators of our policy will be deregistered from the course. Read this to know what is plagiarism.