INDIAN STATISTICAL INSTITUTE

MTech(CS) I year 2020-2021

Subject: Computing Laboratory

Assignment 2

Total: $3 \times 20 = 60$ marks Deadline: 07:00am, 23 February, 2021

SUBMISSION INSTRUCTIONS

1. You may consult course material, or material from other Internet sources; you may also discuss the problems and their possible solutions with your classmates. You are permitted to use any code provided to you as part of the course material, but apart from this exception, **all code must be written by you without looking at anyone else's code**. Please acknowledge all sources that you take help from.
2. You may write your programs in C/C++ or Python.
3. Please make sure that your programs adhere strictly to the specified input and output format.
4. Please adhere to the file naming conventions discussed in class (i.e., your file names should be of the form `cs20XX-assign2-progY` with a `.c` or `.py` extension). Note that your file names **do not** need to contain a 6-8 character long password like alphanumeric string.
5. Please upload your programs to `https://www.dropbox.com/request/pUwjQQIWY1KbhisODb1H`.

Q1. You are given a set of tasks, T_0, T_1, ..., T_{n-1}, along with the time required to complete each task. Each task T_i may have some prerequisites, i.e., other tasks that need to be completed before T_i can be started. Assuming that only one task can be carried out at any given time, write a program to efficiently construct a schedule following which all tasks may be properly completed. Your algorithm should ideally be linear in the size of the input.[1] If multiple schedules are possible, you may print any one.

Input format

For a set of n tasks, the input will consist of $n + 1$ lines. The first line will contain just the number n. The remaining n lines will each contain the duration in seconds of a task T_i ($0 \leq i < n$), and the serial numbers of zero or more tasks which have to be completed before T_i can be started.

Output format

Your program should print n lines, each containing two fields: a start time, and the task which should be started at that time. The start time for the first task in your schedule should always be 0. If no feasible schedule exists, your program should print `No feasible schedule exists`.

Sample input

```
6
2
5
3  0 1 4
```

[1] Think carefully about what this means.

```
7
11 1 3
1  4
```

Sample output

```
0  0
2  1
7  3
14 4
25 5
26 2
```

Q2. Suppose a program has been organized in a country that is run by an eccentric dictator. The program consists of a series of presentations by various persons. Given the initial schedule, the dictator demands that a new schedule be prepared, satisfying the following weird constraints.

- In the revised schedule, no person should have the same position as in the original schedule.
- A person with a longer name should present earlier.
- For cases where the two requirements above are conflicting, the first constraint will have priority.

See the samples below for a more detailed explanation of the above constraints.

Write a program that will take a series of names of presenters (in order of their appearance in the initial plan), and rearrange those names in the best possible way such that the above criteria are satisfied. If multiple correct answers exist, it is sufficient to return any one of those.

Input Format

The input — a series of strings representing the names of presenters — will be provided as command-line arguments.

Output Format

The names of the presenters should be printed (one name per line) on the standard output in the order specified by the revised schedule.

Sample Input 0

```
Bob Michael Elvis
```

Sample Output 0

```
Michael
Elvis
Bob
```

Sample Input 1

```
Bob Michael Elvis Tansen
```

Sample Output 1

```
Michael
Tansen
Bob
Elvis
```

Explanation: Elvis should normally have been the 3rd presenter in the revised schedule, appearing immediately after Tansen. However, he was also the 3rd presenter according to the original schedule. To satisfy the dictator's primary whim, Bob presents before Elvis.

Sample Input 2

```
Tansen Mozart
```

Sample Output 2

```
Mozart
Tansen
```

Q3. Consider an $m \times n$ matrix A consisting of integer entries. Given a particular pair of indices (i, j) as user input, write a program to efficiently find the number of elements that appear after $A[i, j]$ during row/column-major traversal and are no more than it by value.

Input Format

Input will be provided via standard input in the following format. The first line of input will consist of a pair of integers, namely the number of rows (m) and number of columns (n) of the matrix A. This will be followed by the elements of A, providing each row in a single line following their order in the matrix. Finally, in the last line, a pair of indices will be provided along with a character (either `r` or `c`). This character will represent the mode of traversal, namely `r` for row-major and `c` for column-major. Note that, indexing starts with 0.

Output Format

Output is to be printed on the standard output in the following format. The output will print the number of elements appearing after $A[i, j]$ through row/column-major traversal, such that they are less than or equal to $A[i, j]$.

Sample Input 0

```
1 8
-1 10 -5 5 10 15 -3 10
0 2 r
```

Sample Output 0

```
0
```

Sample Input 1

```
3 6
40 29 30 31 26 26
15 25 35 40 41 22
60 20 22 38 40 11
0 2 c
```

Sample Output 1

```
5
```

Sample Input 2

```
4 4
10 10 10 10
10 10 10 10
10 10 10 10
10 10 10 10
1 1 r
```

Sample Output 2

```
10
```