CS 6103D Software Systems Laboratory

PROBLEM 1D Evaluation **15.09.2022**

General Instructions

- The evaluation consists of two parts PART A and PART B. You will be allowed to proceed to PART B ONLY if you complete(partially/fully) PART A and submit the code in EduServer.
- Design:
  - Write the design only for **the function specified in the question paper** and submit it in the **EduServer before 10.45 AM**.
  - Design submitted after 10.45 AM will not be evaluated.
  - No need to write a design for PART B.
- Implementation:
  - Implement **PART A,** make sure that your program works correctly for the given sample I/O and submit code for PART A in the EduServer **before 11.45 AM. If you need more time for completing PART A, you may request your instructor for the same**.
  - After submitting PART A, you may inform the instructor that you have submitted and then proceed with coding for PART B.
  - Submit the source code for **PART B** in the EduServer **before 12.45 PM** and get the result verified by your evaluator before **1.15 PM**.

Mark Distribution:

Maximum Marks - 10 marks

PART A

Design - 2 marks

Viva - 1 mark

Implementation - 3 marks

PART B

Viva - 1 mark

Implementation - 3 marks

Modify the program developed for **problem 1D** as follows:

Part A

To the program you wrote for **problem 1D**, add a function ***getMaxPreRequisitesCourse()*** that prints codes of course(/courses) that have maximum number of direct prerequisites.

Design: Write algorithm (in pseudocode) for the ***getMaxPreRequisitesCourse()*** function.

Input/Output Format

The input should be read from a file '*input.txt*'.

The input file consists of multiple lines.

- The first line contains an integer *n>0*, the number of courses in a semester (or the number of vertices in the DAG).
- The next line contains an integer *m*, the number of edges in the DAG.
- The next *n* lines contain details of the *n* courses - in each line, *code* and *name* of a course, separated by a single space.
- The next set of lines may contain a pair of strings representing course codes, *code1* and *code2,* indicating that course *code1* is a prerequisite for course *code2* (and hence a directed edge from vertex *code1* to vertex *code2*)
- The next set of lines indicate the operations to be performed. Each line contains a character from *{p, e}*.
    - Character *p*:
        - Call function *getMaxPreRequisitesCourse()*
            - If there are more than one course with maximum number of direct prerequisites, print the code of courses in a single line with a space separating the course codes.
    - Character e: Terminate the program.

Sample Input (file *input.txt*)

```
5
4
ZZ1004D Computer Programming
```

CS2002D Program Design

CS2006D Discrete Structures

CS2005D Data Structures and Algorithms

CS3006D Computer Networks

ZZ1004D CS2006D

CS2002D CS2006D

CS2002D CS3006D

CS2005D CS3006D

p

e

Output

CS2006D CS3006D //any order

Part B

To the program you wrote for **problem 1D**, add a function ***getSucceedingCourses(c)*** that prints the code of courses that can be taken by those who completed course c (directly or after completing some additional course/courses)in a topological order, seperated by a space.

Input/Output Format

The input should be read from a file '*input.txt*'.

The input file consists of multiple lines.

- The first line contains an integer *n>0*, the number of courses in a semester (or the number of vertices in the DAG).
- The next line contains an integer *m*, the number of edges in the DAG.
- The next *n* lines contain details of the *n* courses - in each line, *code* and *name* of a course, separated by a single space.
- The next set of lines may contain a pair of strings representing course codes, *code1* and *code2,* indicating that course *code1* is a prerequisite for course *code2* (and hence a directed edge from vertex *code1* to vertex *code2*)
- The next set of lines indicate the operations to be performed. Each line begins with a character from *{t, e}* followed by zero or more string(s).

- Character *t*: Character *t* followed by a string *code* corresponding to the course code.
  - Call function *getSucceedingCourses(code)*
- Character e: Terminate the program.

Sample Input (file input.txt)

5

5

ZZ1004D Computer Programming

CS2002D Program Design

CS2006D Discrete Structures

CS2005D Data Structures and Algorithms

CS3006D Computer Networks

ZZ1004D CS2002D

CS2002D CS2006D

CS2002D CS2005D

CS2006D CS2005D

CS2005D CS3006D

t CS2002D

e

Output

CS2006D CS2005D CS3006D