CS 6103D Software Systems Laboratory

PROBLEM 1C Evaluation **22.10.2021**

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 PART A and submit the code in EduServer.
- Design:
  - Write the design ***only*** for **PART A** and submit it in the EduServer **before 2:20 PM**.
  - 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 3:00 PM**. **If you need more time for completing PART A, you may request your instructor for the same.** But you shall be permitted to proceed to PART B ONLY if you complete the coding for PART A before **3:30 PM** and in that case, you have to complete PART B before **4:00 PM**.
  - After submitting PART A, you may inform the instructor that you have submitted and then proceed with coding for PART B.
  - No need to submit the source code in EduServer for **PART B**. You should complete the coding for PART B before **3:30 PM** and get the result verified by your evaluator before **4:15 PM.**

Mark Distribution:

Maximum Marks - 8 marks

Design - 2 marks
Viva - 2 marks
Implementation - 4 marks (Part A - 2 + Part B - 2)

Modify the program developed for problem 1C as follows:

Part A

To the program you wrote for Problem 1, add a function *getNumRegistered(c)* that returns the number of students registered in the course with course code *c*. Your algorithm should return the count by counting the number of nodes in the *regList* (implemented using BST)

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

Input/Output Format (same as in PS1C except the menu item *p* for printing)

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.
- The next *n* lines contain details of the *n* courses - in each line, *code, name,* and *credits* of a course, separated by a single space.
- The next set of lines indicate the operations to be performed. Each line begins with a character from *{i, d, s, p, e}* followed by zero or more string(s)/ integer(s).
  - Character *i*: Character *i* followed by two strings *stud_name* and *code* corresponding to the student name and course code respectively, separated by a space.
    - Call function *insert(stud_name, t)* to insert a new node with the given *stud_name* to the tree *t* corresponding to the *regList* of the course *code.*
  - Character *d*: Character *d* followed by two strings *stud_name* and *code* corresponding to the student name and course code respectively, separated by a space.
    - Call function *delete(stud_name, t)* to delete the *stud_name* from the tree *t* corresponding to the *regList* of the course *code.*
  - Character *s*: Character *s* followed by two strings *stud_name* and *code* corresponding to the student name and course code respectively, separated by a space.
    - Call function *search(stud_name, t)* to check if the *stud_name* is present in the tree *t* corresponding to the *regList* of the course *code.*
      - If *stud_name* is present in *t*, the function should return a pointer to the node containing *stud_name* and print "Student *stud_name* registered in course *code*"
      - Otherwise, the function should return NIL and print "Student *stud_name* has not registered in course *code*"
  - Character *p*: Character *p* followed by a string *code* corresponding to the course code.
    - In the first line, print the course *code*, *name*, and *credits* (separated by a space) of the given course.
    - In the next line, print "Number of students registered" followed by a space followed by the number of students registered (by invoking function *getNumRegistered(code)*)

- Call function *printRegList(code)* to print the list of students registered, by performing an *Inorder Traversal* of the tree *t,* in a single line and each *stud_name* separated by a space.
- If *regList* is empty, print "No students enrolled for this course".

- Character e: Terminate the program.

Sample Input (file *input.txt*)

```
4
CS6101D MFC 4
CS6111D ALG 4
CS6213D FIS 4
CS6103D SSL 1
i SARITHA CS6103D
i NEHA CS6103D
i ALI CS6213D
p CS6101D
i NEHA CS6213D
i RIA CS6111D
d NEHA CS6213D
p CS6213D
i ALI CS6101D
i SAMEER CS6101D
i SARITHA CS6101D
i ANCY CS6213D
i JOHN CS6213D
i RIA CS6213D
i SARITHA CS6213D
d RIA CS6213D
d SAMEER CS6101D
d ALI CS6213D
s JOHN CS6213D
s SAMEER CS6101D
p CS6101D
p CS6213D
p CS6111D
p CS6103D
e
```

Output

```
CS6101D MFC 4
 No students enrolled for this course
CS6213D FIS 4
```

Number of students registered 1
ALI
Student JOHN registered in course CS6213D
Student SAMEER has not registered in course CS6101D
CS6101D MFC 4
Number of students registered 2
ALI SARITHA
CS6213D FIS 4
Number of students registered 3
ANCY JOHN SARITHA
CS6111D ALG 4
Number of students registered 1
RIA
CS6103D SSL 1
Number of students registered 2
NEHA SARITHA
2

Part B

To the program you wrote for Problem 3, add a function *getPreRequisites(c)* that returns an array of codes of courses that are direct prerequisites for *c*.

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 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, p, e}* followed by zero or more string(s).
  - Character *t*: Display an order in which the courses can be credited, in a single line, with a space separating the course codes.
  - Character *p*: Character *p* followed by a string *code* corresponding to the course code.
    - Call function *getPreRequisites(code)*

- If there are more than one direct prerequisites for the course *code*, print the course codes of the direct prerequisites in a single line with a space separating the course codes.
- Print "There are no prerequisites for course *code*." if there are no direct prerequisites for course *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
p ZZ1004D
p CS2005D
e
```

Output

```
ZZ1004D CS2002D CS2006D CS2005D  CS3006D
There are no prerequisites for course ZZ1004D
CS2002D CS2006D
```