CS 6103D Software Systems Laboratory

PROBLEM 1C Evaluation **13.09.2022**

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 **PART A** in a text file and submit it in the **EduServer before 9.45 AM**.
  - Design submitted after 9.45 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 10.45 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 11.45 AM** and get the result verified by your evaluator before **12.15 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 1C** as follows:

Part A

To the program you wrote for **Question 1**, add a function ***getMaxRegisteredCourse()*** that prints the *code* of course(/courses) registered by maximum number of students. Your algorithm should count the number of nodes in each *regList* (implemented using BST).

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

Input/Output Format

The input 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, m, e}* followed by zero or more string(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 *m*:
    - Call function *getMaxRegisteredCourse()* to print the *code* of course(/courses) registered by maximum number of students, separated by a space.

- Character e: Terminate the program.

Sample Input

4
CS6101D MFC 4
CS6111D ALG 4
CS6213D FIS 4
CS6103D SSL 1
i SARITHA CS6103D
i NEHA CS6103D
i NEHA CS6213D
i RIA CS6111D
m
d NEHA CS6213D
i ALI CS6213D
i SAMEER CS6213D
i SARITHA CS6111D
m
e

Output

CS6103D
CS6103D CS6111D CS6213D <==*(any order)*

Part B

Modify the *course struct* by adding new fields - *type* (string) and *maxLimit* (int). The possible values of the two fields are:

- *type* - either "core" or "elective"
- *maxLimit* - 50 for "core", 3 for "elective".

Modify the program you wrote for **Question 3** such that all students can register for core courses. The *waitList* is required only for elective courses.

Input/Output Format

The input 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, credits* and *type* 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, p, e}* followed by zero or more string(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.
        - If the *type* of the course is 'elective':
            - If the number of students in *regList* corresponding to course *code* is *maxLimit*, create a *student* node with 2 fields *stud_name* and *priority* and insert the new *student* node into the **max priority queue** Q corresponding to the *waitList* of the course *code.*
                - If Q is empty, initialize the *priority* value as *maxLimit* and assign it to the new *student* node *and* call function *insert(Q, student)*
                - *Otherwise,* decrement the current *priority* (requires keeping additional attribute, current priority value for each course) value by 1 and assign it to the new *student* node and call function *insert(Q, student).*

(Assume that the **max-heap property** is maintained based on *priority.)*

- Otherwise, insert a new node with the given *stud_name* into the *regList* corresponding to the course *code.*
- If the *type* of the course code is 'core', insert a new node with the given *stud_name* into *regList* corresponding to the course *code.*

- Character *d*: Character *d* followed by two strings - *stud_name* and *code* corresponding to the student name and course code resp., separated by a space.
  - Delete the node with the given student details from the *regList* of the course *code.*
  - If *waitList* corresponding to the course *code* is not empty, extract the node with highest priority from *waitList* and insert the node to *regList* corresponding to the course *code.*
- Character *p*: Character *p* followed by a string *code* corresponding to the course code.
  - Print the details of the course *code* as follows:
    - in the first line, print the course *code*, *name*, *credits, type,* and *maxLimit* (separated by a space).
    - in the next line print the list of students registered, by performing an *Inorder Traversal* of the tree regList*,* each *stud_name* separated by a space.
    - No need to print waitlisted student names.
- Character e: Terminate the program.

Sample Input

4
CS6101D MFC 4 core
CS6111D ALG 4 core
CS6213D FIS 4 elective
CS6103D SSL 1 core
i NEHA CS6103D
i ALI CS6103D
i SARITHA CS6103D
i RIA CS6103D

p CS6103D
i ALI CS6213D
i RIA CS6213D
i NEHA CS6213D
i JOHN CS6213D
i RAM CS6213D
p CS6213D
d RIA CS6213D
p CS6213D
e

Output

CS6103D SSL 1 core 50
ALI NEHA RIA SARITHA
CS6213D FIS 4 elective 3
ALI NEHA RIA
CS6213D FIS 4 elective 3
ALI JOHN NEHA