CSD201 ASSIGMENT

General Instructions

Read the instructions below carefully before starting the exam.

Allowed Materials and Tools

- Materials on the computer (including JDK, NetBeans)
- For distance learning: Google Meet, Hangout (for exam supervision)
- Software tools to be used: NetBeans IDE 8.x and Java JDK 1.8

Steps to Complete the Exam

1. Create a folder to store project files, e.g., CSD_given (1). Download the provided materials into (1).
2. Complete the exam requirements.
3. Before submission:
    - Clean and Build Project (Shift+F11)
    - Then rename the dist folder to run. (If the run folder already exists, delete it before renaming)
4. Submit your work:
    - To submit project Q1, first select Question number: 1
    - Browse and select the project folder (e.g., Q1, or Q1A, or Q2, ...)
    - Then click the Submit button
5. Additional requirements:
    - Do not use Vietnamese with diacritics when writing comments in the program
    - Do not add new import statements to the provided files

Failure to comply with any of the above requirements will result in a ZERO grade.

Important Notes

- The expected input and output examples are used only to check your code
- The expected input and output in actual test cases (for grading) will be different from the examples below
- Do not hard code with the given expected results

Troubleshooting

If the given project (e.g., Q1) runs with errors, you need to run "Clean and Build Project" (Shift+F11). If it still has errors, try to rename or copy the project to another project, e.g., from Q1 to Q1X or Q1Y.

Student and Scholarship Management System Description

The provided files contain code to implement a student and scholarship management program. The structure of the main classes is as follows:

- **Student Class**: contains information about a student, including id, name, gpa, major, and balance (amount in the account)
- **TreeNode Class**: includes a Student object and left/right pointers for the BST structure
- **ListNode Class**: includes a Student object and a next pointer for linking
- **StudentBST Class**: a binary search tree managing all students in the system
- **ScholarshipList Class**: a linked list managing scholarship recipients
- **StudentManager Class**: the main class of the program, containing StudentBST to store data about existing students in the system and ScholarshipList corresponding to the list of scholarship recipients

Note: In this problem, when a student belongs to the scholarship list, the scholarship amount is stored in the balance field of the Student object.

Students need to carefully read the provided code to fully understand the relationship between classes and the functions in each class. The specific tasks of the exam are to implement the following requirements:

Requirements

f1(): (2.5 points)

To complete the f1 requirement, students need to perform two specific tasks:

- Implement the insert() function in the StudentBST structure - insert a new student into the BST
- Implement the addLast() function in the ScholarshipList structure - add a student to the end of the scholarship list

Expected output to check your code:

Student BST (Inorder Traversal):

(S001,John,3.8,CS,100.0) (S002,Mary,3.5,Business,150.0) (S003,David,3.9,CS,200.0) (S004,Sarah,3.2,IT,120.0) (S005,Michael,3.7,Engineering,180.0)

Scholarship List:

(S003,David,3.9,CS,500.0) (S001,John,3.8,CS,600.0) (S005,Michael,3.7,Engineering,450.0)

Note: The scholarship list has been initialized with sample students from the data.txt file.

f2(): (2.5 points)

To complete the f2 requirement, students need to:

- Implement the findHighestGPA() function in the StudentBST structure to find the student with the highest GPA
- Display this student's information

Expected output to check your code:

Student BST (Inorder Traversal):

(S001,John,3.8,CS,100.0) (S002,Mary,3.5,Business,150.0) (S003,David,3.9,CS,200.0) (S004,Sarah,3.2,IT,120.0) (S005,Michael,3.7,Engineering,180.0)

Scholarship List:

(S003,David,3.9,CS,500.0) (S001,John,3.8,CS,600.0) (S005,Michael,3.7,Engineering,450.0)

Student with highest GPA: (S003,David,3.9,CS,200.0)

f3(): (2.5 points)

To complete the f3 requirement, students need to:

- Count and display the number of students with GPA greater than or equal to 3.5
- Implement the countByGPA() method in the StudentBST structure

Expected output consists of only a single number (the count of students with GPA ≥ 3.5), for example:

4

f4(): (2.5 points)

To complete the f4 requirement, students need to:

- Implement the removeById() function in the ScholarshipList structure to remove a student from the scholarship list
- Implement the updateStudentBalance() function in the StudentBST structure to update the amount in the student's account
- Process scholarships: remove students from the scholarship list and add the scholarship amount to the student's account in the BST

Expected output to check your code:

Student BST (Inorder Traversal):

(S001,John,3.8,CS,100.0) (S002,Mary,3.5,Business,150.0) (S003,David,3.9,CS,200.0) (S004,Sarah,3.2,IT,120.0) (S005,Michael,3.7,Engineering,180.0)

Scholarship List:

(S003,David,3.9,CS,500.0) (S001,John,3.8,CS,600.0) (S005,Michael,3.7,Engineering,450.0)

Student BST (Inorder Traversal):

(S001,John,3.8,CS,700.0) (S002,Mary,3.5,Business,150.0) (S003,David,3.9,CS,700.0) (S004,Sarah,3.2,IT,120.0) (S005,Michael,3.7,Engineering,630.0)

Scholarship List:

Empty