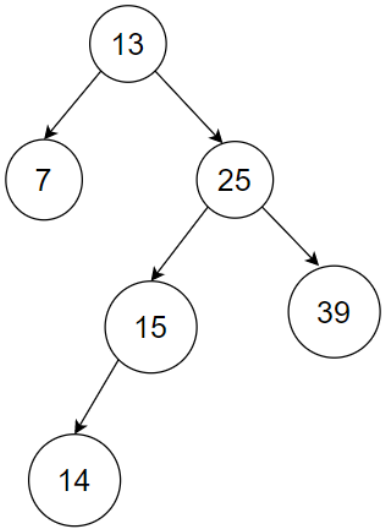
National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Data Structures**	**Course Code:**	**CS 2001**
Program:	**BS(Computer Science)**	**Semester:**	**Fall 2021**
Due Date	**12-Dec-2021 at 11:59 pm**	**Total Marks:**	**40**
Type:	**Assignment 4**		
Sections	**E and F**		

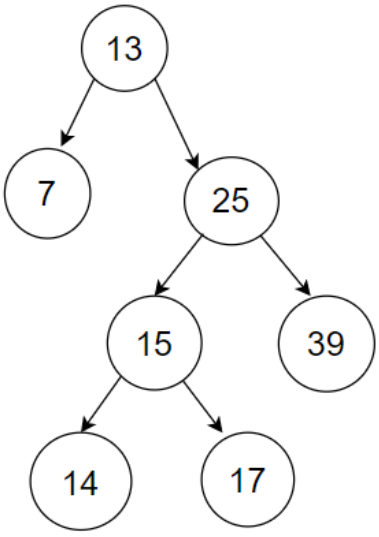
Important Instructions:

1. Submit your source files in a zipped file named as your roll number, i.e., 20L-1111.zip.
2. You are not allowed to copy solutions from other students. We will check your code for plagiarism using plagiarism checkers. If any sort of cheating is found, negative marks will be given to all students involved.
3. Late submission of your solution is not allowed.

Consider a binary search tree given below.



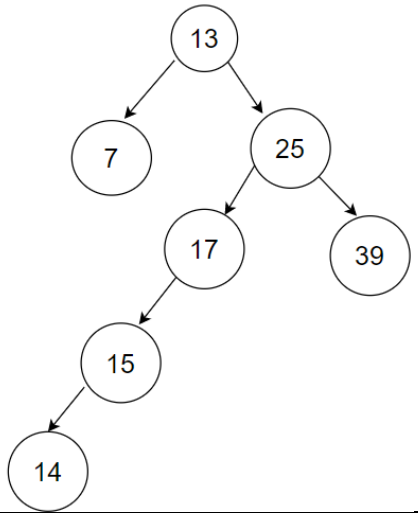
Now if we insert 17 in this tree, the tree will become like this:



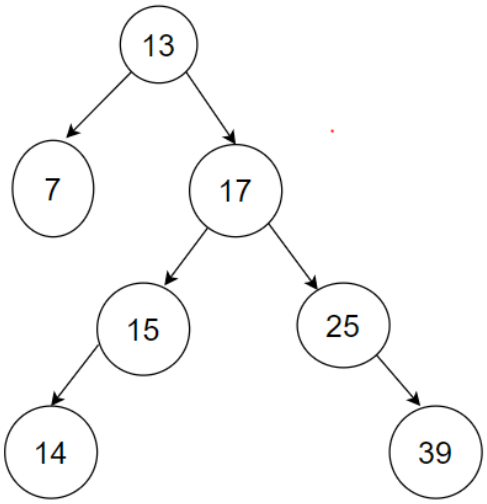
Computer programs tend to follow a phenomenon known as locality of reference. There are several types of this phenomenon, such as temporal locality of reference, spatial locality of reference, etc. Temporal locality of reference states that a program is highly likely to access the same data repetitively over a short period of time.

In our binary search tree, the last accessed data was the node containing the key 17. So, if our program follows temporal locality of reference, it is highly likely to access it again. For example, the program may execute the function: search(17). So, in order to make search fast, we can perform a series of rotations so that the node containing 17 comes at level 0, i.e., it becomes the root node. For this purpose, we will have to perform a number of left rotations and right rotations. In one rotation, the desired node will move one level up. For example:

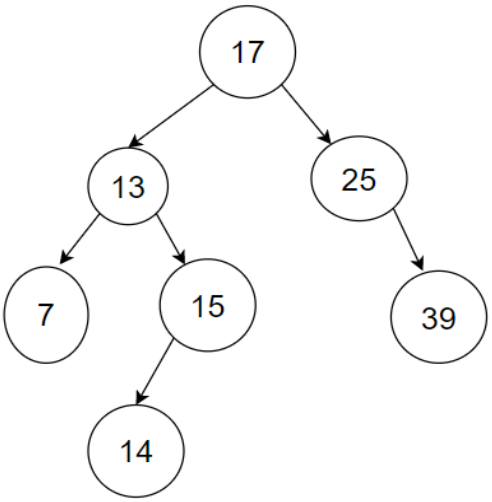
Step 1: The node containing 17 is at level 3. So to move this node to level 2, we will perform left rotation on the node containing 15. The tree after rotation will become:



Step 2: To move 17 to level 1, we need to perform right rotation on the node containing 25. The tree after rotation will become:



Step 3: Now to move 17 to level 0, we need to perform left rotation on the node containing 13. The tree after rotation will become:



Part a: [20 marks]

Now develop a template-based binary search tree (let's call it SpecialBST). Each node in SpecialBST will have a <key, value> pair along with left child and right child pointers. SpecialBST performs a series of rotations so that the last accessed node becomes the root node. In case of insert function, the newly inserted node is the last accessed node. In case of search function, the node containing the desired key is the last accessed node. In case of delete function, the parent of the deleted node is the last accessed node. However, it is possible that we tried to search for a key in our special BST, but the key did not exist. In such a case, the last accessed node is the node where our search was terminated. For example, let's say we searched for key 50 in our special BST (shown in the very first diagram). Since key 50 does not exist, so our search would terminate at 39. Hence, the node containing 39 becomes the last accessed node. Now perform the required rotations, so that 39 becomes the new root. After rotations, the tree will still be a binary search tree. In case we wanted to delete 50, the node containing 39 would still be the last accessed node, so it should be moved to the root via rotations. Your SpecialBST class should have the following member functions:

1. Default constructor: SpecialBST()
2. Insert function: bool insert(K key, V value); returns false if not inserted due to duplicate key, otherwise returns true.
3. Search function: V* search(K key)
4. Delete function: bool delete(K key); returns true if deleted else returns false (key did not exist).
5. Function to get all values in a vector. The values should be placed level-wise, i.e., if we print values present in the vector (from left to right), we get level-order printing: vector<V>* getValuesLevelWise()
6. Function to get all values in a vector such that if we print values present in the vector (from left to right), we get inorder printing: vector<V>* getValuesInOrder()
7. Destructor: ~SpecialBST()

Part b: [10 marks]

Write a class StudentList that internally uses SpecialBST class made in part(a) to store the data of students. The attributes of students that we will store are roll number, first name, last name, batch, department and cgpa. This StudentList class should have the following member functions:

1. bool InsertNewStudent(int rollNumber, string firstName, string lastName, int batch, string department, float cgpa);
2. bool deleteStudent(int rollNumber);
3. bool updateStudent(int oldRollNumber, int newRollNumber, string newFirstName, string newLastName, int newBatch, string newDepartment, float newCgpa);
4. void printAllStudents(); //prints the students in rollnumber-wise sorted order.

You can also add any other member functions in this class if you need. In update student function, if we are not changing the roll number, then updation is straightforward. However, if we are also changing the roll number, then you have to think about how we can do it. Changing the roll number also requires checking for duplicate roll numbers.

Part c: [10 marks]

Now write a main program that uses the class made in part (b) and shows a menu. A sample run of this part is shown below (> indicates user-entered input):

Press **I** to insert a new student.
Press **D** to delete a student.
Press **S** to search a student by roll number.
Press **U** to update the data of a student.
Press **P** to print all students sorted by roll number.
Press **E** to exit.

> I
Please enter the data of new student in the order: Roll Number, First Name, Last Name, Batch, Department, CGPA
> 29 Hammad Tariq 2020 CV 2.50
New student inserted successfully!

Press **I** to insert a new student.
Press **D** to delete a student.
Press **S** to search a student by roll number.
Press **U** to update the data of a student.
Press **P** to print all students sorted by roll number.
Press **E** to exit.

> I
Please enter the data of new student in the order: Roll Number, First Name, Last Name, Batch, Department, CGPA
> 29 Rasheed Khan 2017 MG 3.10
Roll Number already exists!

Press **I** to insert a new student.
Press **D** to delete a student.
Press **S** to search a student by roll number.
Press **U** to update the data of a student.
Press **P** to print all students sorted by roll number.
Press **E** to exit.

> I
Please enter the data of new student in the order: Roll Number, First Name, Last Name, Batch, Department, CGPA
> 1 Zaheer Abbas 2020 CS 2.90
New student inserted successfully!

Press **I** to insert a new student.
Press **D** to delete a student.
Press **S** to search a student by roll number.
Press **U** to update the data of a student.
Press **P** to print all students sorted by roll number.
Press **E** to exit.

> P
Total Students: 2

Roll Number: 1
First Name: Zaheer
Last Name: Abbas
Batch: 2020
Department: CS
CGPA: 2.90

Roll Number:29
First Name:Hammad
Last Name: Tariq
Batch: 2020
Department: CV
CGPA: 2.50

Press **I** to insert a new student.
Press **D** to delete a student.
Press **S** to search a student by roll number.
Press **U** to update the data of a student.
Press **P** to print all students sorted by roll number.
Press **E** to exit.

> S
Please enter the roll number of the student that you want to search:
> 1
Roll Number: 1
First Name: Zaheer
Last Name: Abbas
Batch: 2020
Department: CS
CGPA: 2.90

Press **I** to insert a new student.
Press **D** to delete a student.
Press **S** to search a student by roll number.
Press **U** to update the data of a student.
Press **P** to print all students sorted by roll number.
Press **E** to exit.

> U
Please enter the roll number of the student whose data you want to update:
> 1
Please enter new data of student in this order: Roll Number, First Name, Last Name, Batch, Department, CGPA
> 1 Zaheer Abbas 2020 EE 3.90
Student data updated successfully.

Press **I** to insert a new student.
Press **D** to delete a student.
Press **S** to search a student by roll number.
Press **U** to update the data of a student.

Press **P** to print all students sorted by roll number.
Press **E** to exit.

>D
Please enter the roll number of the student that you want to delete:
>1
Student deleted successfully.

You should handle certain incorrect inputs, such as non-positive roll number and year, incomplete input data, etc.