CS29003 Algorithms Laboratory
Assignment 5: Play with Sorting and BST

General instruction to be followed strictly

1. Do not use any global variable unless you are explicitly instructed so.
2. Do not use Standard Template Library (STL) of C++.
3. Use proper indentation in your code and comment.
4. **Name your file as** `<roll_no>_<assignment_no>`**. For example, if your roll number is 14CS10001 and you are submitting assignment 3, then name your file as** `14CS10001_3.c` **or** `14CS10001_3.cpp` **as applicable.**
5. **Write your name, roll number, and assignment number at the beginning of your program.**
6. Make your program as efficient as possible. Follow best practices of programming.
7. Submit your program on Moodle before deadline. Submissions by email or any other means will NOT be considered for evaluation.

In this assignment we will learn an awesome sorting algorithm. The idea is as follows. Take any balanced BST, say AVL tree. Insert the keys to be sorted one by one in this tree. All insertions together taked $\mathcal{O}(n \log n)$ time. To output the sorted order of the keys, simply perform appropriate traversal of the BST. Total time complexity is $\mathcal{O}(n \log n)$. On top of that, we can also add/delete elements from our set in $\mathcal{O}(\log n)$ time while being able to output the keys in sorted order whenever asked in $\mathcal{O}(n)$ which we will need anyway to simply write n keys. Let us now apply this idea in the following situation.

We now write a program to efficiently maintain the students' records in IIT Kharagur. Suppose we store the name, roll number, and CGPA of each student in our record. Let us assume that each name contains at most 99 characters, each roll number is an integer between 1 and 10000, and each CGPA is a floating point number in $[0, 10]$. Use the following struct data type to store the record of each student.

```
typedef struct{
        char name[100];
        int roll;
        float cgpa;
}record;
```

We want to display all student records in sorted order, both ascending and descending, with respect to name, roll, and CGPA each in $\mathcal{O}(n \log n)$ time (refer to sample output below). We also want to support search with respect to either name or roll or CGPA in $\mathcal{O}(\log n)$ time. Also adding/deleting a new record should be done in $\mathcal{O}(\log n)$ time.

We achieve the above performance requirement by maintaining three AVL (or any height balanced BST) trees each for name, roll, and CGPA. Use the following struct data type for the AVL tree for name.

```
#define MAX 100
typedef struct _node_name{
        char name[MAX];
        struct _node_name *left,*right;
        record *data; //points to the record corresponding to the name
        char balance; //balance = height of left subtree - height of right subtree
}node_name;
```

Note that each node of the AVL tree stores a variable called “balance” which is the height of the left subtree minus the height of the right subtree. This variable becomes useful to restoring balance in AVL tree after an insert or delete option. In an AVl tree, this balance variable will store either 0 or 1 or -1 and hence char data type which is one byte, long is enough.

Use similar struct data types for roll and CGPA. To insert a new record, create a record with all the information and insert its address in all the three AVL trees. Perform the delete operation similarly (think when you should free the memory held by the deleted record). For searching, perform search of the key in the appropriate AVL tree to get the pointer to the record. You can assume that all the names, rolls, and CGPAs are distinct. The CGPA should be printed up to two decimal places.

Submit a single .c or .cpp file. Your code should get compiled properly by gcc or g++ compiler.

You can refer to the slides on AVL tree which is available on Moodle. All the best!

Sample Output

```
palash@palash-ThinkPad-X1-Yoga-3rd:~$ ./a.out
1. insert
2. delete by name
3. search by name
4. search by roll
5. search by CGPA
6. Print ascending by name
7. Print descending by name
8. Print ascending by roll
9. Print descending by roll
10. Print ascending by CGPA
11. Print descending by CGPA
12. Exit
Choose option: 1
Write name: Rahul
Write roll: 10
Write CGPA: 8.62
Choose option: 1
Write name: Kabir
Write roll: 15
Write CGPA: 8.4
Choose option: 1
Write name: John
Write roll: 90
Write CGPA: 7.98
Choose option: 7
Descending by name: (Rahul, 10, 8.62),(Kabir,15,8.40),(John,90,7.98),
Choose option: 10
Ascending by CGPA: (John,90,7.98),(Kabir,15,8.40),(Rahul, 10, 8.62),
Choose option: 3
Write name: Vijay
No record found for Vijay
Choose option: 5
Write CGPA: 8.43
(Kabir,15,8.43)
Choose option: 2
Write name: Rahul
(Rahul, 10, 8.62) deleted
Choose option: 9
Descending by roll: (John,90,7.98),(Kabir,15,8.40),
Choose option: 12
Program exits
```

Policy on Plagiarism

Academic integrity is expected from all the students. You should work on the assignment/exam consulting only the material we share with you. You are required to properly mention/cite anything else you look at. Any student submitting plagiarized code will be penalized heavily. Repeated violators of our policy will be deregistered from the course. Read this to know what is plagiarism.