National Institute of Technology Calicut
Department of Computer Science and Engineering
Fourth Semester B. Tech.(CSE)-Winter 2021-22
CS2094D Data Structures Laboratory
Assignment #3 - Part A

Submission deadline (on or before): 18.02.2022, 9:00 AM

Policies for Submission and Evaluation:

- You must submit your assignment in the Eduserver course page, on or before the submission deadline.
- Ensure that your programs will compile and execute without errors using gcc compiler.
- During the evaluation, failure to execute programs without compilation errors may lead to zero marks for that evaluation.
- Your submission will also be tested for plagiarism, by automated tools. In case your code fails to pass the test, you will be straightaway awarded zero marks for this assignment and considered by the examiner for awarding F grade in the course. Detection of ANY malpractice related to the lab course can lead to awarding an F grade in the course.

Naming Conventions for Submission

- Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar, .tar, .gz). The name of this file must be

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>.zip`

  (Example: *ASSG1_BxxyyyyCS_LAXMAN.zip*). DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.
- The source codes must be named as

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c`

  (For example: *ASSG1_BxxyyyyCS_LAXMAN_1.c*). If you do not conform to the above naming conventions, your submission might not be recognized by our automated tools, and hence will lead to a score of 0 marks for the submission. So, make sure that you follow the naming conventions.

Standard of Conduct

- Violation of academic integrity will be severely penalized. Each student is expected to adhere to high standards of ethical conduct, especially those related to cheating and plagiarism. Any submitted work MUST BE an individual effort. Any academic dishonesty will result in zero marks in the corresponding exam or evaluation and will be reported to the department council for record keeping and for permission to assign F grade in the course. The department policy on academic integrity can be found at: http://cse.nitc.ac.in/sites/default/files/Academic-Integrity_new.pdf.

QUESTIONS

1. Write a program to create an AVL TREE A and perform the operations *insertion*, *deletion*, *search* and *traversal*. Assume that the AVL TREE A does not contain duplicate values. Your program should contain the following functions.

   - INSERT(A, k) – Inserts a new node with key 'k' into the tree A.
   - SEARCH(A, k) - Searches for a node with key 'k' in A, and returns a pointer to the node with key k if one exists; otherwise, it returns NIL.
   - DELETENODE(A, k) – Deletes a node with the key 'k' from the tree A.
   - GETBALANCE(A, k) – Prints the balance factor of the node with 'k' as key in the tree A.

     Note:- Balance factor is an integer which is calculated for each node as:

     $$B\_factor = height(left\_subtree) - height(right\_subtree)$$

   - LEFTROTATE(A, k) – Perform left rotation in the tree A, with respect to the node with key 'k'.
   - RIGHTROTATE(A, k) – Perform right rotation in the tree A, with respect to node with key 'k'.
   - PRINTTREE(A) – Prints the tree given by A in the paranthesis format as: (t (left-subtree)(right-subtree)). Empty parentheses ( ) represents a null tree.

     Note: After each insertion on an AVL TREE, it may result in increasing the height of the tree. Similarly, after each deletion on an AVL TREE, it may result in decreasing the height of the tree. To maintain height balanced property of AVL tree, we may need to call rotation functions.

   Input Format:

   - Each line contains a character from '*i*', '*d*', '*s*', '*b*', '*p*' and '*e*' followed by at most one integer. The integers, if given, are in the range $[-10^6, 10^6]$.
   - Character '*i*' is followed by an integer separated by space; a node with this integer as key is created and inserted into A.
   - Character '*d*' is followed by an integer separated by space; the node with this integer as key is deleted from A and the deleted node's key is printed.
   - Character *'s'* is followed by an integer separated by space; find the node with this integer as key in A.
   - Character '*b*' is followed by an integer separated by space; find the balance factor of the node with this integer as key in A and the print the balance-factor.
   - Character '*p*' is to print the PARENTHESIS REPRESENTATION of the tree A.
   - Character *'e'* is to 'exit' from the program.

   Output Format:

   - The output (if any) of each command should be printed on a separate line.
   - For option '*d*', print the deleted node's key. If a node with the input key is not present in A, then print FALSE.
   - For option '*s*', if the key is present in A, then print TRUE. If key is not present in A, then print FALSE.
   - For option '*b*', if the key k is present in A, then print the balance factor of the node with k as key. If key is not present in A, then print FALSE.
   - For option *'p'*, print the PARENTHESIS REPRESENTATION of the tree A.

Sample Input:

```
i 4
i 6
i 3
i 2
i 1
s 2
p
b 4
d 3
p
e
```

Sample Output:

```
TRUE
( 4 ( 2 ( 1 ( ) ( ) ) ( 3 ( ) ( ) ) ) ( 6 ( ) ( ) ) )
1
3
( 4 ( 2 ( 1 ( ) ( ) ) ( ) ) ( 6 ( ) ( ) ) )
```

2. Write a program to check whether a BST is an AVL tree or not. You are given a set of integers that represents an arrangement in which elements are inserted in a BST. First construct a BST from the given input and each node in the tree consists of a key and two pointers (left and right).

```
struct node {
    int key;
    struct node* left;
    struct node* right;
}
```

Do not alter the structure of the tree (should use only one tree). Your program should include the following functions.

- **isAVL**(struct node* root): returns 1 if the tree is an AVL tree otherwise 0.

Input Format:

- Each line contains a character '*i*' followed by an integer separated by space; a node with this integer as key is created and inserted into the BST.
- For option *'c'*, check the tree is AVL or not.
- Character *'t'* is to 'terminate' the program.

Output Format:

- Print 1 if the tree is an AVL tree otherwise print 0.

Sample Input 1:

```
i 10
i 15
i 20
c
t
```

Sample Output 1:

```
0
```

Sample Input 2:

```
i 37
```

```
i 21
i 80
c
t
```

Sample Output 2:

```
1
```