Indian Institute of Technology Roorkee

Department of Computer Science and Engineering

CSN-261: Data Structures Laboratory (Autumn 2019-2020)

Lab Assignment-3 (L3) **Date: August 7, 2019** **Duration: 2 Weeks**

General Instructions:

1. Every Lab Assignment will be performed by the students individually. No group formation is required and the evaluations will be done every week for the students individually.
2. Use of Linux OS is mandatory for this Lab to complete all the Lab Assignments.
3. The total execution time or CPU time is to be reported for all the programs in each Lab Assignment.
4. The student should use Deoxygen tool (http://www.doxygen.nl) for getting automatic documentation of the codes written by him/her.
5. For version control of the written codes, the students are being instructed to learn and use any open-source CSV tool (https://www.nongnu.org/cvs) or GitHub (https://github.com).

Submission and Evaluation Instructions:

1. **Submit your** zipped folder (**<filename>.zip** or **<filename>.tar.gz**) through your account in Moodle through the submission link for this Lab Assignment in Moodle course site: https://moodle.iitr.ac.in/course/view.php?id=46.
2. **Hard deadline for Final submission in Moodle: Aug 21, 2019 (1:00 pm Indian Time).** For any submission after Final Deadline, 20% marks will be deducted (irrespective of it is delayed by a few seconds or a few days). The key to success is starting early. You can always take a break, if you finish early.
3. The submitted zipped folder (**<filename>.zip** or **<filename>.tar.gz**) must contain the following:
   (a) The source code files in a folder
   (b) A report file (**<filename>.DOC** or **<filename>.PDF**) should contain the details like:
      i. Title page with details of the student
      ii. Problem statements
      iii. Algorithms and data structures used in the implementation
      iv. Snapshots of running the codes for each problem
4. The submission by each student will be checked with others' submission to identify any copy case (using such detection software). If we detect that the code submitted by a student is a copy (partially or fully) of other's code, then the total marks obtained by one student will be divided by the total number of students sharing the same code.

Instructions for L3:

1. Objective of this Lab Assignment L3 is to make the students familiar with different data structures while coding the programs in the C language to solve some real-life problems.
2. The students are expected to have a basic knowledge of data structures and the C programming language.
3. It is mandatory to learn and use gdb (GNU Debugger) for debugging the programs in Linux platform after installing gdb (https://www.gnu.org/software/gdb). There is also an interesting online GDB tool to learn: https://www.onlinegdb.com.
4. The student will have to demonstrate and explain the coding done for this Lab Assignment L3 in the next laboratory class to be held on **Aug 21, 2019** for evaluation.

Problem 1:
Given the set of integers, write a C++ program to create a binary search tree (BST) and print all possible paths for it. You are not allowed to use subarray to print the paths.

Convert the obtained BST into the corresponding AVL tree for the same input. AVL tree is a self-balancing binary search tree. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property.

Convert the obtained BST into the corresponding red-black tree for the same input. Red-Black Tree is a self-balancing Binary Search Tree (BST) where every node follows following rules.
1) Every node has a color either red or black.
2) Root of tree is always black.
3) There are no two adjacent red nodes (A red node cannot have a red parent or red child).
4) Every path from a node (including root) to any of its descendant NULL node has the same number of black nodes.

Write a menu driven program as follows:
1. To insert a node in the BST and in the red-black tree
2. To create AVL tree from the inorder traversal of the BST
3. To print the inorder traversal of the BST/AVL/red-black tree
4. To display all the paths in the BST/AVL tree/red-black tree
5. To print the BST/AVL tree/red-black Tree in the terminal using level-wise indentation (print color for red-black tree)
6. Exit

Example:

Input:
10 20 30 40 50 25

Output:

BST:
```
10 [4]
        20 [3]
                25
                30 [2]
                        40 [1]
                                50
```
Inorder traversal (This output is for BST Tree): 10 20 25 30 40 50

AVL Tree:
```
30 [0]
        20 [0]
                10
                25
        40 [1]
                50
```
Inorder traversal (This output is for AVL Tree): 10 20 25 30 40 50

Red Black Tree:
```
20 [2] [BLACK]
        10 [BLACK]
        40 [1] [RED]
```

30 [1] [BLACK]
25 [RED]
50 [BLACK]

Inorder traversal (This output is for Red Black Tree): 10 20 25 30 40 50

Paths (This output is for AVL tree):
30->20->10
20->10
10
30->20->25
20->25
25
30->40->50
40->50
50

Problem 2:
For a given sequence of positive integers A_1, A_2, …, A_N in decimal, find the triples (i, j, k), such that $1 \le i < j \le k \le N$ and $A_i \oplus A_{i+1} \oplus \ldots \oplus A_{j-1} = A_j \oplus A_{j+1} \oplus \ldots \oplus A_k$, where $\oplus$ denotes bitwise XOR. This problem should be solved using dynamic programming approach and linked list data structures.

Input:
(a) Number of positive integers N.
(b) N space-separated integers A_1, A_2, …, A_N.

Output:
Print the number (count) of triples and list all the triplets in lexicographic order (each triplet in a new line).

Example:
Input:
N = 3
5 2 7

Output:
2
(1, 2, 3)
(1, 3, 3)