National Institute of Technology Calicut
Department of Computer Science and Engineering
Fourth Semester B. Tech.(CSE)-Winter 2022-23
CS2094D Data Structures Laboratory
Assignment Cycle 2 Part A

Submission deadline (on or before): 15.02.2024, 11:00 PM

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

  `ASSGC<NUMBER>_<PART>_<ROLLNO>_<BATCHNO>_<FIRST-NAME>.zip`

  (Example: $ASSGC2\_A\_BxxyyyyCS\_CS01\_LAXMAN.zip$). 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

  `ASSGC<NUMBER>_<PART>_<ROLLNO>_<BATCHNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c`

  (For example: $ASSGC2\_A\_BxxyyyyCS\_CS01\_LAXMAN\_1.c$). 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: https://minerva.nitc.ac.in/?q=node/650.

QUESTIONS

1. You are given a sequence of operations to perform on an AVL tree. Each operation is represented by a character followed by its respective parameters. The operations are as follows: [i , x , p , s , e]

   Input format:

   - Each line contains a character from '*i*', '*x*', '*p*', '*s*', '*e*' followed by zero or one integers. The integers, if given, are in the range $\in [1, 10^6]$.
   - i: AVL_insert(T, key) - Insert the given key into the AVL tree T.
   - x: AVL_find(T, key) - Search for the given key in the AVL tree T. If the key is found, print the path from the root to the node containing the key; otherwise, print -1.
   - p: AVL_preorder(T) - Print the preorder traversal of the AVL tree T.
   - s: AVL_cal(T) - Given a point, calculate the total number of left and right rotations required to balance the AVL tree T at that point. left rotation, right rotation separated by space
   - e: Exit - Terminate the sequence of operations.

   NOTE : $\boldsymbol{key} \in [1, 10^6]$

   Output format:

   For

   - x : sequence of integers with space separated them representing path
   - p : sequence of integers with space separated them representing preorder.
   - s : two integer representing total number of left_rotaions and right_rotaions

   Sample Input 1:

   ```
   4
   i 10
   i 20
   i 30
   s
   i 40
   p
   i 50
   i 5
   i 1
   s
   x 10
   x 56
   e
   ```

   Sample Output 1:

   ```
    1 0
    20 10 30 40
   2 1
   20 5 10
   -1
   ```

   Sample Input 2:

   ```
   i 167
   i 34
   i 121
   ```

s
i 56
i 198
i 73
i 112
p
x 112
s
e

Sample Output 2:

1 1
121 56 34 73 112 167 198
121 56 73 112
2 1

2. You are tasked with implementing a data structure that mimics the functionality of an ordered map using an AVL Tree. An ordered map, also known as a map or dictionary, is a data structure that stores a collection of key-value pairs. Unlike a regular hash map, an ordered map maintains the keys in sorted order, allowing efficient retrieval of elements in a sorted sequence based on their keys. Your main() function shall read the choice from the console and call the following functions appropriately:

   (a) insert(AVLTree *root, key, value): Inserts a key-value pair into the AVL Tree. If the key already exists, update the corresponding value.

   (b) lower_bound(AVLTree *root, key): Prints the key-value pair of the first element with not less than the given key, otherwise print -1.

   (c) find(AVLTree *root, key): Searches for a key in the AVL Tree and print the key-value pair if found, otherwise print -1.

   (d) size(AVLTree *root): Prints the number of key-value pairs in the AVL Tree.(0 if empty).

   (e) empty(AVLTree *root):Prints 1 if the AVLTree is empty, otherwise prints 0.

   (f) printElements(AVLTree *root): Prints the keys in the AVL Tree in ascending order, (-1 if empty).

Input Format:

All the inputs in a line are separated by space.
Each line of input starts with a character from the menu list ['i','l','f','s','e','p','t'] and may be followed by one or two integers, which specify the key(and value), respectively.

- Input 'i' followed by two integers key and value calls the function insert(AVLTree *root, key, value).
- Input 'l' followed by an integer key calls the function lower_bound(AVLTree *root, key).
- Input 'f' followed by an integer key calls the function find(AVLTree *root, key).
- Input 's' calls the function size(AVLTree *root).
- Input 'e' calls the function empty(AVLTree *root).
- Input 'p' calls the function printElements(AVLTree *root).
- Input 't' terminates the execution of the program.

Output Format:

- The output of the result of any menu is printed in a new line with a space between each integer.
- A line may contain -1.

Sample Input1:

```
i 1 10
i 2 20
i 3 30
i 4 40
l 3
l 5
f 35
e
f 1
s
p
i 1 100
i 10 1
f 1
s
p
t
```

Sample Output1:

```
3 30
-1
-1
0
1 10
4
1 2 3 4
1 100
5
1 2 3 4 10
```

Sample input 2:

```
e
i 5 50
i 3 30
i 7 70
i 1 10
l 6
f 3
i 9 99
s
p
e
f 10
t
```

Sample output 2:

```
1
7 70
3 30
5
1 3 5 7 9
0
-1
```

3. Write a program to Create a Red Black Tree from the given input. Your program should include the following function:

   - InsertinRedBlack(struct Node* root,key): Inserts a new node with the 'key' into the tree and prints parenthesized representation (with corresponding colors) of the created red-black tree.

   NOTE: A Red-Black tree is a self-balancing binary search tree where every node obeys the following rules:

   - Every node is either red or black.
   - The root is always black.
   - There are no two adjacent red nodes (A red node cannot have a red parent or red child).
   - All paths from a node to descendant nodes contain the same number of black nodes.

   Input Format:

   - Each line of the input contains a positive integer 'key' or a character 'e'.
   - If the input is a positive integer then Call function InsertinRedBlack(root, key).
   - If 'e' is encountered, terminate the program.

   Output format:

   - For each line of the input, the corresponding line of the output should contain the Parenthesis Representation (key value followed by the color) of the current tree.

   Sample Input 1:

   15
   8
   27
   34
   4
   63
   22
   e

   Sample output 1:

   (15 B ( ) ( ))
   (15 B (8 R ( ) ( )) ( ))
   (15 B (8 R ( ) ( )) (27 R ( ) ( )))
   (15 B (8 B ( ) ( )) (27 B ( ) (34 R ( ) ( ))))
   (15 B (8 B(4 R ( ) ( )) ( )) (27 B ( ) (34 R ( ) ( ))))
   (15 B (8 B(4 R ( ) ( )) ( )) (34 B (27 R ( ) ( )) (63 R ( ) ( ))))
   (15 B (8 B (4 R ( ) ( )) ( )) (34 B (27 B (22 R ( ) ( )) ( )) (63 B ( ) ( ))))

   Sample input 2:

   10
   4
   15
   2
   5
   e

Sample output 2:

(10 B ( ) ( ))
(10 B (4 R ( ) ( )) ( ))
(10 B (4 R ( ) ( )) (15 R ( ) ( )))
(10 B (4 B (2 R ( ) ( )) ( )) (15 B ( ) ( )))
(10 B (4 B (2 R ( ) ( )) (5 R ( ) ( ))) (15 B ( ) ( )))

4. Given a parenthesis representation of a tree. Construct the tree for the given input and check whether the tree satisfies the red-black tree properties or not. Write a function "checkRedblack-Tree(root)" to check the tree is red-black tree or not and return return boolean values 1 if it is a red-black tree otherwise 0.

Input format:

- First line of the input contains a space separated Paranthesis representention of the tree T. R -¿red and B-¿black

Output format:

- Return the value "1" if it is a red-black tree ,otherwise return "0"

Sample input 1:
(10 B (4 B (2 R ( ) ( )) (5 R ( ) ( ))) (15 B ( ) ( )))
Sample output 1:
1

Sample input 2:
(15 B (8 B (4 R ( ) ( )) ( )) (34 B (27 R (22 B ( ) ( )) ( )) (63 B ( ) ( ))))
Sample output 2:
0