National Institute of Technology Calicut
Department of Computer Science and Engineering
Third Semester B. Tech.(CSE)-Monsoon 2024
CS2091E Data Structures Laboratory
Assignment 2

Submission deadline (on or before): 21/08/2024, 11:59 PM

Policies for Submission and Evaluation:

- Assignment 2 consists of Part A and Part B (Part B will be uploaded to the Eduserver course page on 14/08/2024), and you **must submit all the solutions of Part A and Part B** combined into a single zip file following the below mentioned guidelines in the Eduserver course page on or before the specified deadline.
- Ensure that your programs will compile and execute using GCC compiler without errors. The programs should be compiled and executed in the SSL/NSL.
- 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: https://minerva.nitc.ac.in/?q=node/650.

Part A

1. Given a sequence of operations to perform on an AVL tree T with n keys, where $n \in [1, 10^6]$. Implement a menu driven program with the following functions:

   (a) $AVL\_insert(T, n)$- Insert the given key n into the AVL tree T. Perform necessary rotations to balance the AVL tree and AVL tree properties are maintained after the insertion.

   (b) $AVL\_find(T, n)$- Search for the given key n in the AVL tree T.

   (c) $AVL\_postorder(T)$- Perform postorder traversal and print the keys in the AVL tree T.

   (d) $AVL\_rotations(T)$- Calculate and print the total number of left and right rotations required so far to balance the AVL tree T from the beginning.

   (e) $AVL\_delete(T, n)$- Print all the ancestors of n in T (order starting from the node itself up to the root) and delete the given key n. Ancestor of a node n: We call any node y on the unique simple path from the root of the tree to n, an ancestor of n. Please note that n is also an ancestor of itself. For the delete operation, if required, replace a node/key with its *inorder successor*.

   (f) $AVL\_balanceFactor(T, n)$- Find the balance factor of the given node n in T.

   Input format:

   - Each line contains a character from { 'i', 'f', 'p', 's', 'd', 'b', 'e' } followed by zero or one integer n, where $n \in [1, 10^6]$.
   - Character 'i' is followed by a positive integer n separated by a space. Perform $AVL\_insert(T, n)$ operation. Assume that unique values are inserted.
   - Character 'f' followed by a positive integer n separated by a space. Perform $AVL\_find(T, n)$ operation.
   - Character 'p' is to perform postorder traversal of T, $AVL\_postorder(T)$.
   - Character 's' Perform $AVL\_rotations(T)$ operation.
   - Character 'd' followed by a positive integer n separated by a space. Perform $AVL\_delete(T, n)$ operation.
   - Character 'b' followed by an integer n separated by a space. Perform $AVL\_balanceFactor(T, n)$ operation.
   - Character 'e' is to terminate the sequence of operations.

   Output Format

   - The output (if any) of each command should be printed on a separate line. However, no output is printed for 'i'and 'e'.
   - For option 'f', If the given key is found, print the path from the root to the node containing the keys separated by a space; otherwise, print -1.
   - For option 'p', print the keys in the nodes of T obtained from the postorder traversal. Sequence of integers are separated by a space.
   - For option 's', print two integers representing total number of left_rotations followed by the total number of right_rotations separated by a space.
   - For option 'd', print all the ancestors of the deleted node n in T. The output is printed as a space-separated sequence in a single line. Print -1 if the key n is not present.
   - For option 'b', print balance factor of the given node n in T. Print -1 if the key n is not present.

Sample test case 1

Input:

```
i 10
i 20
i 30
s
i 40
p
i 50
i 5
b 20
i 1
s
f 10
f 56
d 40
e
```

Output:

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

2. You are asked to implement a data structure that mimics the functionality of an *ordered map* K using an AVL Tree.

   - **Ordered map** K
     - 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 elements in such a way that the **keys** in sorted order, allowing efficient retrieval of elements in a sorted sequence based on their keys.

   Implement the following Operations:

   (a) $Main()$: Read the choice from the console and call the following functions appropriately:

   (b) $Insert(K, key, value)$: Inserts a *key-value* pair into the ordered map K.

   (c) $UpperBound(K, key)$: Among all those elements in K having the *key* not less than the given *key*, print the *key-value* pair of the one with the minimum key.

   (d) $Find(K, key)$: Searches for a *key* in the ordered map K.

   (e) $Size(K)$: Prints the number of *key-value* pairs in ordered map K.

   (f) $Empty(K)$: Check whether the ordered map is empty or not K.

   (g) $DisplyElements(K)$: Display all the *keys* in the ordered map K in descending order.

Input format:

- Each line contains a character from { 'i', 'u', 'f', 's', 'e', 'd', 't' } followed by one or two positive integers.
- Character 'i' followed by two positive integers *key* and *value* separated by a space, calls the function $insert(K, key, value)$. if the entered *key* already exists, update its corresponding *value*.
- Character 'u' followed by a positive integer *key* calls the function $UpperBound(K, key)$.
- Character 'f' followed by a positive integer *key* calls the function $Find(K, key)$.
- Character 's' calls the function $Size(K)$.
- Character 'e' calls the function $Empty(K)$.
- Character 'd' calls the function $DisplyElements(K)$.
- Character 't' terminates the execution of the program.

Output Format

- The output (if any) of each command should be printed on a separate line. However, no output is printed for 'i' and 't'.
- For option 'u', on calling $UpperBound(K, key)$, if such elements are present then print the *key-value* pair of the first element with not less than the given *key* separated by a space, otherwise print -1.
- For option 'f', if the entered *key* is found then print the *key-value* pair separated by a space, otherwise print -1.
- For option 's', print the number of *key-value* pairs in the ordered map or print 0 if K is empty.
- For option 'e', Prints 1 if the ordered map is empty, otherwise prints 0.
- For option 'd', Prints the *key*s in K in descending order separated by a space, print -1 if the ordered map is empty.

Sample test case 1
Input:

```
e
i 1 10
i 2 20
i 3 30
i 4 40
u 3
u 5
f 3
f 1
s
d
i 1 100
i 10 1
f 1
s
d
e
t
```

Output:

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