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

Submission deadline (on or before): 07/08/2024, 11:59 PM

Policies for Submission and Evaluation:

- You must submit the solutions of **Part B** of this assignment following the below-mentioned guidelines in the Eduserver course page, on or before the submission deadline. **Part A** is meant for practice which you are supposed to complete before proceeding to Part B.
- 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

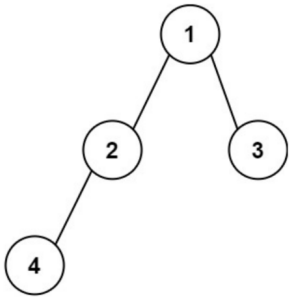
- Violation of academic integrity will be severely penalized. Each student is expected to adhere to high standards of ethical conduct, especially those related to cheating and plagiarism. Any submitted work MUST BE an individual effort. Any academic dishonesty will result in zero marks in the corresponding 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

A **PARENTHESIS REPRESENTATION** of a binary tree is a string consisting of parentheses and integers (separated by spaces) representing a binary tree. The representation should be based on a preorder traversal of the binary tree and must adhere to the following guidelines:

- Each node in the tree should be represented by its integer value.
- Parentheses for Children: For every node (either left or right), its children should be represented inside parentheses. Specifically:
  - If a node has a left child, the value of the left child should be enclosed in parentheses immediately following the node's value.
  - If a node has a right child, the value of the right child should also be enclosed in parentheses. The parentheses for the right child should follow those of the left child.
- Empty parentheses are in the string when a node has an empty left or right child, except for leaf nodes.

For example, consider the given binary tree




```
Parentheses
representation : 1(2(4())(3)
```

1. Given a binary search tree T, implemented using pointers and self-referential structures, and an integer key K, find the successor of K in T.
2. Given a binary search tree T, implemented using pointers and self-referential structures, and an integer key K, find the predecessor of K in T.
3. Given a binary search tree T, implemented using pointers and self-referential structures, and an integer key K, print the sub-tree rooted at K using parenthesis representation.
4. Given the parenthesis representation of a binary tree T, convert it to a linked representation (implemented using pointers and self-referential structures).
5. Given the parenthesis representation of a tree T, check whether T is a valid binary tree or not.
6. Given the parenthesis representation of a binary tree T, check whether T is complete or not.
7. Given the parenthesis representation of a binary tree T, count the number of leaf nodes in T.
8. Given the parenthesis representation of a binary tree T, check whether T is a binary search tree (BST) or not.

Part B

1. Given an inorder and a preorder traversals of a binary tree, construct the unique binary tree T corresponding to these traversals. The binary tree contains non-negative integers.

   The program should include the following functions:

   (a) $main()$: Repeatedly reads an input character from the menu list through the terminal and executes menu driven operations accordingly.

   (b) $PostOrder(T)$: Prints the post order traversal of the binary tree T.

   (c) $ZIG\_ZAG(T)$: Prints the *Zig zag traversal* (Definition is given below) of the binary tree T.

   (d) $LevelMax(T)$: Prints the nodes with maximum value at each level of the binary tree T.

   (e) $Diameter(T)$: Prints the diameter (Definition is given below) of the binary tree T.

   (f) $RightLeafSum(T)$: Prints the sum of the values of *right leaf nodes*(Definition is given below).

   Definitions:

   - **Depth of a node**: The length (number of edges) of the unique simple path from the root to the node.
   - **Level of a tree**: Level i (i varies from 0 to max depth) of a binary tree contains all nodes at depth i (level 0 contains only root node).
   - **Zig zag Traversal**: Level-order traversal of the binary tree where nodes are printed in a zig-zag pattern in a single line. Specifically, at even levels (where level numbering starts from 0), nodes are printed from right to left and at odd levels, nodes are printed from left to right.
   - **Diameter of a tree**: The diameter/width of a tree is defined as the number of nodes on the longest simple path between any two leaf nodes.
   - **Right Leaf Node**: A leaf node that is present as a right child to its parent node.

   Input Format:

   - The first line contains an integer $n \in [1, 10^6]$ indicating the number of nodes in the tree.
   - The second line contains a space-separated sequence of n integers representing the $INORDER$ traversal of the tree T with key values $\in [1, 10^6]$.
   - The third line contains a space-separated sequence of n integers representing the $PREORDER$ traversal of the tree T with key values $\in [1, 10^6]$.
   - Each subsequent line contains a character from the set {'p', 'z', 'm', 'd', 's', 'e'}.
     - Character 'p' calls $Postorder(T)$ - to print the postorder traversal of the tree.
     - Character 'z' calls $ZIG\_ZAG(T)$ - to print the *Zig zag traversal* of the tree.
     - Character 'm' calls $LevelMax(T)$ - to print the nodes with maximum value at each level.
     - Character 'd' calls $Diameter(T)$ - to print the *diameter* of the binary tree.
     - Character 's' calls $RightLeafSum(T)$ - to print the sum of the values of all *right leaf nodes*.
     - Input 'e' terminates the execution of the program.

   Output Format:

   - The output of each command should be printed on a separate line.
   - For option 'p', print the postorder traversal of T. Each node's value is separated by a space.

- For option 'z', print the *Zig zag traversal* of T. Each node's value is separated by a space.
- For option 'm', print the nodes with the maximum value at each level. Each node's value is separated by a space.
- For option 'd', print the diameter of T.
- For option 's', print the sum of the values of all right leaf nodes.

Test Case_1

Input:

```
5
4 2 1 3 5
1 2 4 3 5
p
z
m
d
s
e
```

Output:

```
4 2 5 3 1
1 2 3 5 4
1 3 5
5
5
```

Test Case_2

Input:

```
7
4 2 5 1 6 3 7
1 2 4 5 3 6 7
p
z
m
d
s
e
```

Output:

```
4 5 2 6 7 3 1
1 2 3 7 6 5 4
1 3 7
5
12
```

2. We are using a Binary Search Tree (BST) to store car details for a car showroom, where each car is represented by its model number, model name, and price. The BST is created based on the model number, which is unique for each car.

   The BST data structure should support various dynamic-set operations and should be implemented with the following specifications:

   - **BST Structure**
     Each node in the BST contains a model number (positive integer), model name (string), and price (positive integer). In addition to these key attributes, each node contains pointers to its left child, right child, and parent. If a child or the parent is missing, the appropriate pointer should be set to `NIL`. The root node is the only node in the tree whose parent is `NIL`.

   - **BST Property**
     Let x be a node in a BST. If y is a node in the left subtree of x, then $y.key < x.key$. If y is a node in the right subtree of x, then $y.key \geq x.key$.

   - **Operations**
     (a) $Main()$: Creates an empty BST and repeatedly reads a character from the console to perform the following operations until 'e' (exit) is entered:
     - 'a' : Add a new car detail.
     - 'd' : Delete an existing car detail.
     - 's' : Search for a car by model number.
     - 'i' : Perform an inorder traversal of the BST and print car details.
     - 'p' : Perform a preorder traversal of the BST and print car details.
     - 't' : Perform a postorder traversal of the BST and print car details.
     - 'm' : Modify the price of an existing car.

     (b) $Create\_Node(model\_number, model\_name, price)$: Creates a new node with the given car details and returns a pointer to the new node. All pointer attributes of the new node should be set to NIL.

     (c) $Add(T, x)$: Inserts the node x into the BST T. Here x is a pointer to the new node returned by the $Create\_Node()$ function.

     (d) $Delete(T, x)$: Deletes the node x from the BST T. Here x is a pointer to the node to be deleted returned by the $Search()$ function to locate the node x.

     (e) $Search(T, model\_number)$: Searches for a node with the given model number in T and returns a pointer to the node if it exists; otherwise, it returns NIL.

     (f) $Inorder(T)$: Performs recursive inorder traversal of the BST T and prints the car details in the nodes of T in inorder.

     (g) $Preorder(T)$: Performs recursive preorder traversal of the BST T and prints the car details in the nodes of T in preorder.

(h) $Postorder(T)$: Performs recursive postorder traversal of the BST T and prints the car details in the nodes of T in postorder.

(i) $Modify(T, model\_number, new\_price)$: Modifies the price of a car with the given model number to the new price.

Input Format

- Each line contains a character from {'a', 'd', 's', 'i', 'p', 't', 'm', 'e'} followed by the arguments required for the corresponding operation.
  - Character 'a' is followed by a positive integer n, a string s, and a positive integer p separated by a space. Perform $Add(T, x)$ operation.
  - Character 'd' is followed by a positive integer n separated by a space. Perform $Delete(T, x)$ operation
  - Character 's' is followed by a positive integer n separated by a space. Perform $Search(T, model\_number)$ operation.
  - Character 'i' is to perform inorder traversal of T.
  - Character 'p' is to perform preorder traversal of T.
  - Character 't' is to perform postorder traversal of T.
  - Character 'm' is followed by two positive integers separated by a space. Perform $Modify(T, model\_number, new\_price)$ operation.
  - 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 car's details in the order: model number, model name, price separated by a space. If a node with the entered model number is not present in T, then print -1.
- For option 's', If a node with entered model number is present in T, then print all the details of that car in the order: model number, model name, price separated by a space. If a node with the entered model number is not present in T, then print -1.
- For options 'i', 'p', 't', print the data in the nodes of T obtained from the corresponding traversal. Each car's details are written on a separate line in the order: model number, model name, price separated by a space.
- For option 'm', print the modified car's details in the order: model number, model name, and updated price separated by a space. If a node with the entered model number is not present in T, then print -1.

Test Case

Input:

```
a 1 Toyota 20000
a 2 Honda 18000
a 3 Ford 25000
s 2
```

```
i
p
t
m 2 19000
s 2
d 3
s 3
i
e
```

Output:

```
2 Honda 18000
1 Toyota 20000
2 Honda 18000
3 Ford 25000
2 Honda 18000
1 Toyota 20000
3 Ford 25000
1 Toyota 20000
3 Ford 25000
2 Honda 18000
2 Honda 19000
2 Honda 19000
3 Ford 25000
-1
1 Toyota 20000
2 Honda 19000
```