TK 1143 Program Design

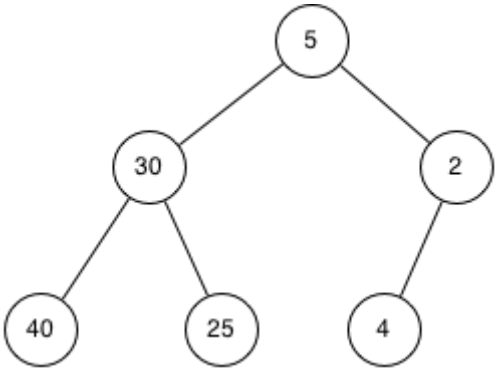
TREE

Section A:

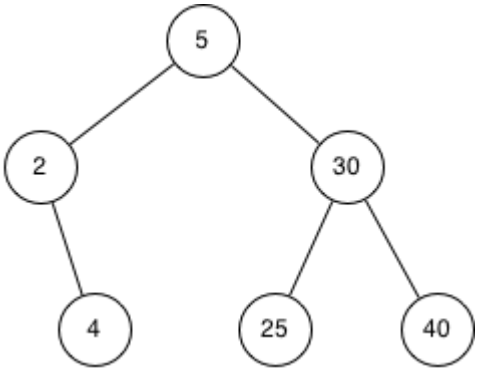
Instruction: Answer all the following questions in the space given.

1. Compare both of the following trees A and B.

A



B



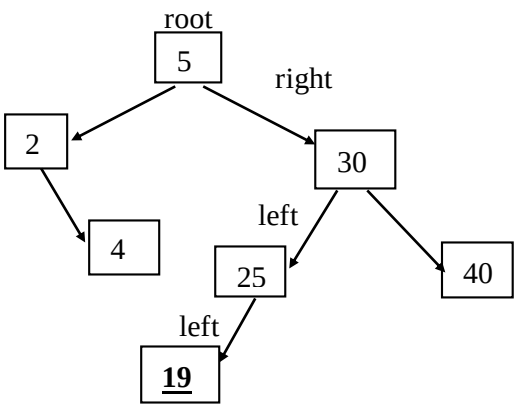
a) Which of the trees shows a correct binary search tree (BST) given that the keys were inserted in the following order 5, 30, 2, 40, 25, 4. Justify your answer.

The correct binary search tree(BST) is B because data key of left child is smaller than its parent and data key of right child is greater than its parent.

b) Identify root, parent node, child node and leaf node based on answer (3a).

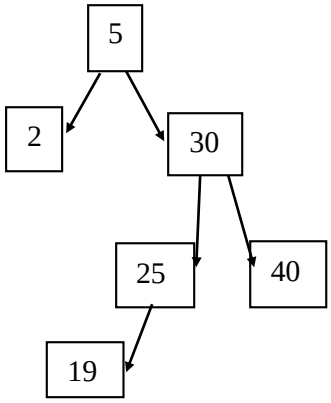
root = 5
parent node = 5 ,30, 2
child node = 2 , 30 , 40 , 25 , 4
leaf node = 4 , 25 , 40

c) Based on answer (3a), illustrates the process for inserting a new node with key 19 into a binary search tree. Justify your answer.

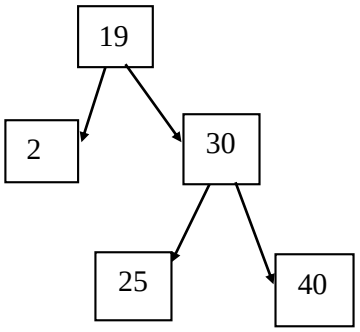


19 is greater than 5 so it will goes to right, 19 is less than 30 so it will goes to left , 19 is less than 25 so it will goes to left and become a child node of node 25.

d) Illustrates the process for deleting node with key 4 followed by key 5 into a current binary search tree on answer (3c). Justify your answer.

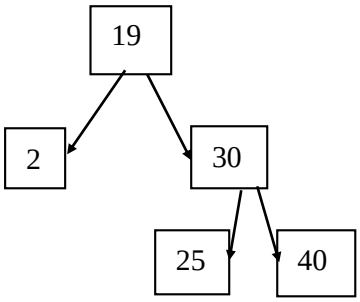


When node 4 is removed, node 2 will not have any node child



When node 4 and 5 is removed, 19 will become a new root because the tree will be rearranged. To determine the other nodes, it will be in order. Node 2 will go to the left because it less than 19 and node 30, 25, 40 will be on the right

e) Illustrates the process for searching node with key 40 and 4 into a current binary search tree on answer (3d). Justify your answer.



To find the key, the first key 40 will be compare with the root which is 19, key 40 is greater than 19 so it will goes to the right . then 40 will compare with 30 , 40 is greater than 30 so it will goes to right and finally found the key 40. It will return true. The second key which is 4 will not be found because key 4 is not exist in the binary search tree so it will return false.

f) Based on current binary search tree on answer (3e), find inorder, preorder, postorder and level order traversals. Tips: Inorder (LNR), Preorder (NLR), Postorder (LRN)

Inorder : 2, 19, 25, 30, 40
Preorder : 19, 2, 30, 25, 40
Postorder : 2, 25, 30, 40, 19

Section B: Answer all the following questions in the space given.

1. Consider the following code.

```
1  public class BinarySearchTree {
2      /* Class containing left and right child of current node and key value*/
3  
4  
5  
6  
7  
8  
9  
10 
11     }
12     // Root of BST
13     Node root;
14 
15     // Constructor
16     BinarySearchTree() {
17         root = null;
18     }
19 
20     // This method mainly calls insertRec()
21     void insert(int key) {
22         root = insertRec(root, key);
23     }
24     // A recursive function to insert a new key in BST
25     Node insertRec(Node root, int key) {
26         if (root == null) {
27             root = new Node(key);
28             return root;
29         }
30         if (key < root.key)
31             root.left = insertRec(root.left, key);
32         else if (key > root.key)
33             root.right = insertRec(root.right, key);
34         return root;
35     }
36 
37     // This method mainly calls InorderRec()
38     void inorder() {
39         inorderRec(root);
40     }
41     //A utility function to do inorder traversal of BST
42     void inorderRec(Node root) {
43         if (root != null) {
44             inorderRec(root.left);
45             System.out.print(root.key + " ");
46             inorderRec(root.right);
47         }
48     }
49 
50     // Driver Program to test above functions
51     public static void main(String[] args) {
52 
53 
54 
55 
56 
57 
58 
59 
60         // print inorder traversal of the BST
61         System.out.println("Inorder traversal of the given tree");
62 
63     }
64 }
```

a. Complete the class Node in Line 3-10 containing left and right child of current node and key value.

```
class Node{
        int key;
        Node left,right;

        public Node(int item){
                key = item;
                left = right = null;
        }
}
```

b. Declare the tree object in Line 53.
BinarySearchTree bst = new BinarySearchTree();

c. Line21-35 is a method to insert new key in BST

d. Line 38-48 is a method to do Inorder reversal of BST

e. Complete the given space in Line 54-59 and 62 based on this input data 15,8,6,12,21,25.

```
tree.insert(15);
tree.insert(8);
tree.insert(6);
tree.insert(12);
tree.insert(21);
tree.insert(25);
```

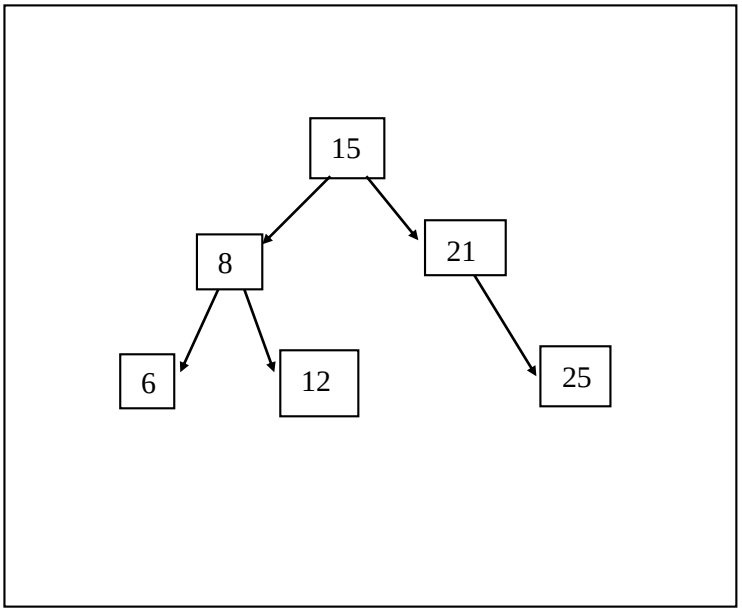
f. Run the code and state the output.

Inorder traversal of the given tree

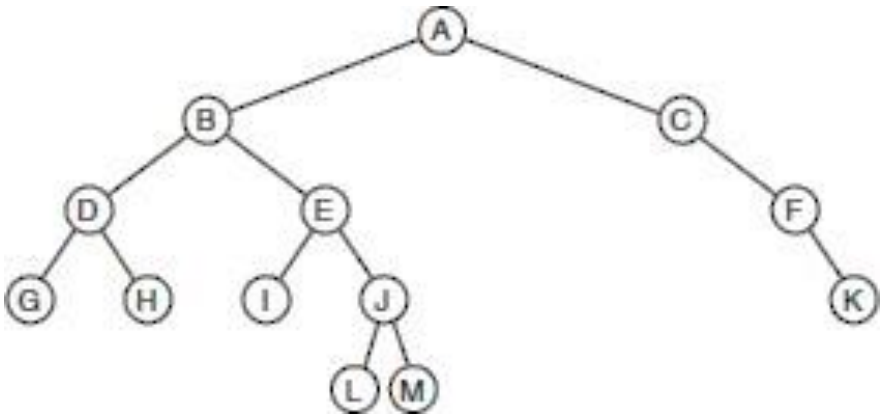
6 8 12 15 21 25

g. Draw the Binary Search Tree (BST).

Hints: Tree where its node can have a max of 2 children. Value of left child< than its parent, AND value of right child > than its parent.



2. Consider the following tree.



a. Is the tree above fulfill the requirement of BST? Justify your answer.

-No because not all alphabet that greater than root are place in the right side of the root which is A based on ASCII

b. Write the algorithm and code for method inorder, preorder and postorder traversal. Refer https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/

Method	Algorithm	Code

Inorder	1. Traverse the left subtree
	2. Visit the root
	3. Traverse the right subtree
		void printInorder(Node node){
		if(node ==null)
		return;
		printInorder(node.left);
		System.out.print(node.key + " ");
		printInorder(node.right);
		}

Preorder	1. Visit the root
	2. Traverse the left subtree
	3. Traverse the right sub tree
		void printPreorder(Node node){
		if(node==null)
		return;
		System.out.print(node.key + " ");
		printPreorder(node.left);
		printPreorder(node.right);
		}

Postorder	1. Traverse the left subtree
	2. Traverse the right subtree
	3. Visit the root
		void printPostorder(Node node){
		if(node==null)
		return;
		printPostorder(node.left);
		printPostorder(node.right);
		System.out.print(node.key + " ");
		}

c. Determine inorder, preorder and postorder

Inorder : G, D, H, B, I, E, L, J, M, A, C, F, K
Pre-order : A, B, D, G, H, E, I, J, L, M, C, F, K
Post-order : G, H, D, I, L, M, J, E, B, K, F, C, A

d. Complete the following main method in the space given.

```
public static void main(String[] args)
{
    BinaryTree tree = new BinaryTree();
    tree.root = new Node("A");
    tree.root.left = new Node("B");
    tree.root.right = new Node("C");
    tree.root.left.left = new Node("D");
    tree.root.left.right = new Node("E");
    tree.root.right.right = new Node("F");
    tree.root.left.left.left = new Node ("G");
    tree.root.left.left.right = new Node ("H");
    tree.root.left.right.left = new Node ("I");
    tree.root.left.right.right = new Node ("J");
    tree.root.right.right.right = new Node ("K");
    tree.root.left.right.right.left = new Node ("L");
    tree.root.left.right.right.right = new Node ("M");

    System.out.println("Inorder traversal of binary tree is ");
    tree.printInorder();
    System.out.println("\nPreorder traversal of binary tree is ");
    tree.printPreorder();
    System.out.println("\nPostorder traversal of binary tree is ");
    tree.printPostorder();
}
```

3. Your task is to create an application that can store five words in tree and delete one word by user input. Display the output in inorder word and modified inorder after deleting the word.
    - Create class Node
    - Declare root of BST
    - Create class constructor
    - Create method of insert, delete and inorder.
    - Read five string word from the user input.
    - Display the output of inorder.
    - Delete one string word from user input.
    - Display the output of modified inorder.

Input	Output
please stay at home covid19 Inorder: at covid19 home please stay Delete: covid19	Inorder: at covid19 home please stay Modified Inorder: at home please stay

```
class BST {
	class Node {
		int key;
		Node left, right;
		public Node(int item) {
			key = item;
			left = right = null;
		}
	}

	Node root; // Root of BST

	BST() { // Constructor
		root = null;
	}

	void deleteKey(int key) { // This method mainly calls deleteRec()
		root = deleteRec(root, key);
	}

	// A recursive function to delete a node in BST
	Node deleteRec(Node root, int key) {
		if (root == null) return root; // Base Case: If the tree is empty
		/* Otherwise, recur down the tree */
		if (key < root.key)
			root.left = deleteRec(root.left, key);
		else if (key > root.key)
			root.right = deleteRec(root.right, key);
		// if key is same as root's key, then This is the node to be deleted
		else {
			// node with only one child or no child
			if (root.left == null)
				return root.right;
			else if (root.right == null)
				return root.left;
			// node with two children: Get the inorder successor
			//(smallest in the right subtree)
			root.key = minValue(root.right);
			// Delete the inorder successor
			root.right = deleteRec(root.right, root.key);
		}
		return root;
	}

	int minValue(Node root)
	{
		int minv = root.key;
		while (root.left != null)
		{
			minv = root.left.key;
			root = root.left;
		}
		return minv;
	}

	// This method mainly calls insertRec()
	void insert(int key) {
```

```
		root = insertRec(root, key);
	}

	// A recursive function to insert a new key in BST
	Node insertRec(Node root, int key) {
		if (root == null) // If the tree is empty, return a new node
		{
			root = new Node(key);
			return root;
		}
		/* Otherwise, recur down the tree */
		if (key == root.key)
			System.out.println("\nThe " + key + " is in the BST");
		if (key < root.key)
			root.left = insertRec(root.left, key);
		else if (key > root.key)
			root.right = insertRec(root.right, key);
		return root; // return the (unchanged) node pointer
	}

	// This method mainly calls printInorder()
	void inorder() {
		printInorder(root);
	}

	// A utility function to do inorder traversal of BST
	void printInorder(Node root) {
		if (root != null)
		{
			printInorder(root.left);
			System.out.print(root.key + " ");
			printInorder(root.right);
		}
	}

	// This method mainly calls searchRec()
	boolean search(int key) {
		root = searchRec(root, key);
		if (root!=null)
			return true;
		else
			return false;
	}

	// This method is to Seracha Rec
	public Node searchRec(Node root, int key)
	{
		// Base Cases: root is null or key is present at root
		if (root==null || root.key==key)
			return root;
		// Key is greater than root's key
		if (root.key < key)
			return searchRec(root.right, key);
		// Key is smaller than root's key
		return searchRec(root.left, key);
	}
}


import java.util.Scanner;

class InorderTree {

	// Driver Program to test above functions
	public static void main(String[] args) {
```

```
		stringBST tree = new stringBST();
		Scanner sc = new Scanner(System.in);
		String input = sc.nextLine();
		String [] words = input.split("\\W+");
		for (String str : words){
			str = str.trim();
			tree.insert(str);
		}

		// print inorder traversal of the BST
		System.out.print("Inorder: ");
		tree.printInorder();

		System.out.print("\nDelete: ");
	tree.deleteKey(sc.next());

		System.out.print("Modified Inorder: " );
	tree.printInorder();
	}
}
```

Section C :

There are four (5) questions given in this section. Your task is to write the program to implement data structure Tree (BST) for each question.

FIND ELEMENT	
Input	Standard Input
Output	Standard Output
Data Structure	BST

Problem Description

Write a Java program that reads integers numbers and end with 0. Your program will need to assign all integers into BST. Your code then will need to search one integer by user input.

Input

Input of this program is integers number that end with 0. The next integer represent the integer that will need to search in the tree.

Output

The output of the program will display either the number is FOUND or NOT FOUND in the tree such as the following sample input-output.

Sample input – output

Input	Output
7 3 9 2 4 8 10 1 5 6 0 2	Number 2 is FOUND in this tree.
7 3 9 2 4 8 10 1 5 6 0 12	Number 12 is NOT FOUND in this tree.

*** Note: Students will need to write the main program and modify the BST class given to fulfill the question requirement.*

MINIMUM AND MAXIMUM ELEMENT

Input	Standard Input
Output	Standard Output
Data Structure	BST

Problem Description

Write a Java program that reads integers numbers and end with 0. Your program will need to assign all integers into BST. Your code then will need to find the minimum and maximum element in the tree.

Input

Input of this program is integers number that end with 0.

Output

Output of the program will display all the elements in the tree using preorder traversal, inorder traversal and postorder traversal followed by the minimum and maximum element in the tree.

Sample input – output

Input	Output
12 3 5 36 2 1 8 0	Preorder : 12 3 2 1 5 8 36 Inorder : 1 2 3 5 8 12 36 Postorder : 1 2 8 5 3 36 12 Minimum Element in tree is : 1 Maximum Element in tree is : 36
-1 3 5 6 23 30 18 0	Preorder : -1 3 5 6 23 18 30 Inorder : -1 3 5 6 18 23 30 Postorder : 18 30 23 6 5 3 -1 Minimum Element in tree is : -1 Maximum Element in tree is : 30

*** Note: Students will need to write the main program and modify the BST class given to fulfill the question requirement.*

TREE WITH PARENTHESES

Input	Standard Input
Output	Standard Output
Data Structure	BST

Problem Description

Write a Java program that reads a passage and assign all words into BST. Your code then will print all the words in parentheses format as below:

(root, (sub-left tree) (sub-right tree))

Input

Input of this program is a passage. A passage consists of N words and symbols. Symbols that will beconsidered in the passage are full stop (.), comma (,), question mark (?) and exclamation mark (!). The passage will have M unique words, where the M is less than or equal to N.

Output

Output of the program is all the words in parentheses format.

Sample input – output

Input	Output
E C A D G I.	(E(C(A()())(D()()))(G()(I()())))
i go to school by bus. the bus is big.	(i(go(by(bus(big()())())())())(to(school(is()())(the()()))()))

*** Note: Students will need to write the main program and modify the BST class given to fulfill the question requirement.*

LIST OF UNIQUE WORDS USING BST

Input	Standard Input
Output	Standard Output
Data Structure	BST

Problem Description

List of Unique Words problem aims to display all words from a given passage in a non-recurring form.Your task is to write a Java program that using Binary Search Tree Class for the **List of Unique Words**.

Input

Input of this program is a passage. A passage consists of N words and symbols. Symbols that will beconsidered in the passage are full stop (.), comma (,), question mark (?) and exclamation mark (!). The passage will have M unique words, where the M is less than or equal to N.

Output

Output of the program is M lines, where each line contains a unique word.

Sample input – output

Input	**Output**
I go to school by bus. The bus is big. The school is also big. I like big school and big bus.	I The also and big bus by go is like school to

*** Note: Students will need to write the main program and modify the BST class given to fulfill the question requirement.*

Solution

The algorithm for List of Unique Words :

Read a passage
Remove unnecessary characters
For all words in the passage:

Search the word in BST
If not found

Add the word in BST

Traverse to display all words using in order traversal.

Full structure of worked-example program: List of Unique Words using BST.

```
1  import java.util.Scanner;
2  public class UniqueWordTree {
3          public static void main(String[] args) {
4
5                          BinarySearchTreeWord1 tree = new BinarySearchTreeWord1();
6                          Scanner in = new Scanner(System.in);
7
8                          String passage = in.nextLine();  // read input passage
9                          String delims = "\\W+"; // split any non word
10                         String [] words = passage.split(delims);
11                         for (String str : words){
12                                 //add code to add word into tree
13                         }
14                         //add code to print all words
15                 }
16
17
```

Tutorial :

1. Write program segment to print the number of repeated words from BST.
2. Discuss the different between the **list of words** and **list of unique words** using BST.

WORD FREQUENCIES

Input	Standard Input
Output	Standard Output
Data Structure	Binary Search Tree

Problem Description

Write a JAVA program that will read all words from a passage and store them in BST. The programthen will display a menu and perform the following task:

- **Delete a word**. If word exists, update the word frequency otherwise print word not exist.
- **Search a word**. If word exists, print the word and its frequency otherwise print not exist.
- **Add a new word**. If the word is in the passage, update its frequency.
- **Print all words** and its frequency in ascending order.

Input

Input of this program is a passage. A passage consists of N words and symbols. Symbols that will beconsidered in the passage are full stop (.), comma (,), question mark (?) and exclamation mark (!). The passage will have M unique words, where the M is less than or equal to N.

Followed by the input code and the required data as specified in the sample input-output.

Sample input-output

Input	Output
Stay at home, stay safe and stay healthy. Practice social distancing at work and at home. Help reduce the risk of infection and preventing covid19. Together we break the chain of covid19. #kita jaga kita. Hapuskan covid19!. 2 pendemic 2 covid19 2 we 1 pendemic 4 1 social 4 3 home 3	Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 2 Enter word to be deleted: pendemic The word pendemic is not in the passage. Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 2 Enter word to be deleted: covid19

work 3 good 2 work 2 social 4 10 99 5	One of the words covid19 was successfully deleted. Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 2 Enter word to be deleted: we One of the words we was successfully deleted. Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 1 Enter new word: pendemic The word pendemic has been added successfully. Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 4 Hapuskan(1) Help(1) Practice(1) Stay(1) Together(1) and(3) at(3) break(1) chain(1) covid19(2) distancing(1) healthy(1) home(2) infection(1) jaga(1) kita(2) of(2) pendemic(1) preventing(1) reduce(1) risk(1) safe(1) social(1) stay(2) the(2) work(1)

	Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 1 Enter new word: social The word social has been added successfully. Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 4 Hapuskan(1) Help(1) Practice(1) Stay(1) Together(1) and(3) at(3) break(1) chain(1) covid19(2) distancing(1) healthy(1) home(2) infection(1) jaga(1) kita(2) of(2) pendemic(1) preventing(1) reduce(1) risk(1) safe(1) social(2) stay(2) the(2) work(1) Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 3 Input Search text: home The word home is used 2 times. Menu: 1. Add new word.

	2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 3 Input Search text: work The word work is used 1 times. Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 3 Input Search text: good The word good is not in the passage. Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 2 Enter word to be deleted: work One of the words work was successfully deleted. Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 2 Enter word to be deleted: social One of the words social was successfully deleted. Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 4 Hapuskan(1) Help(1) Practice(1) Stay(1) Together(1) and(3) at(3) break(1)

	chain(1) covid19(2) distancing(1) healthy(1) home(2) infection(1) jaga(1) kita(2) of(2) pendemic(1) preventing(1) reduce(1) risk(1) safe(1) social(1) stay(2) the(2) Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 10 Invalid code, Enter code 1-5. Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 99 Invalid code, Enter code 1-5. Menu: 1. Add new word. 2. Delete word. 3. Search word. 4. Print all words and its frequency. 5. Exit. Input code: 5 Bye..

*** Note: Students will need to write the main program and modify the BST class given to fulfill the question requirement.*

Solution

The algorithm for Word Frequencies :

Read a passage
Remove unnecessary characters
For all words in the passage

Travers to find the word from the BST
If not found

add the word into the BST and with frequency = 1
else

Traverse to display all words with its frequency using in order traversal.