

record keeping and for permission to assign an F grade in the course. The department policy on academic integrity can be found at:

http://minerva.nitc.ac.in/cse/sites/default/files/attachments/news/Academic-Integrity_new.pdf.

Assignment Questions

1. Write a program to create a Binary Search Tree (BST). Your program should include the following functions.

main() - Repeatedly read one of the strings “insr”, “srch”, “minm”, “maxm”, “pred”, “succ”, “delt”, “inor”, “prer” and “post” from the terminal and call corresponding subfunctions until the string “stop” is read.

insert(*tree*, *element*) – adds the node specified by *element* (which contains the data) into the BST specified by *tree*. **If the element already exists in the tree, insert it to the left subtree.**

search(*tree*, *element*) - Search of the data specified by *element* in the tree *tree*. If found, return a pointer to the corresponding node; otherwise, return NULL. If there are multiple nodes that contain *element*, return that node which has the smallest level.

level(*tree*, *node*) - Find the node *node* in the tree *tree*. If found, return the level of node in the tree. Otherwise, return -1.

findMin(*tree*) – returns the smallest data in the BST specified by *tree*.

findMax(*tree*) – returns the largest data in the BST specified by *tree*.

predecessor(*tree*, *node*) – Find and return a pointer to the inorder-predecessor of the node specified by *node* in the BST specified by *tree*. Return -1, if the element exists in the tree, but there is no inorder-predecessor for the node. Return NULL, if the data is not present in the tree.

successor(*tree*, *node*) – Find and return a pointer to the inorder-successor of the node specified by *node* in the BST specified by *tree*. Return -1, if the element exists in the tree, but there is no inorder-successor for the data. Return NULL, if the data is not present in the tree.

delete(*tree*, *node*) – Find and remove the node specified by *node* from the BST specified by *tree*. If not found, return NULL. **Replace the deleted node with its inorder successor if exists otherwise choose inorder predecessor.**

inorder(*tree*) – does a recursive inorder traversal of the BST.

preorder(*tree*) – does a recursive preorder traversal of the BST.

postorder(*tree*) – does a recursive postorder traversal of the BST.

Input format

Each line of the input contains one of the following:

1. The string “**insr**” followed by an integer *i* : Call function **insert(*tree*, *i*)**
2. The String “**srch**” followed by an integer *k* : Search of a node with key *k*. If found, print its level, otherwise print **-1** (assume **root** had level **0**).
3. The String “**minm**” : Print the output of **findMin(*tree*)**. Print “NIL” if the BST is empty.
4. The String “**maxm**” :Print the output of **findMax(*tree*)**. Print “NIL” if the BST is empty.
5. The String “**pred**” followed by an integer *i*: print the output of **predecessor(*tree*, *i*)**
6. The String “**succ**” followed by an integer *i* : print the output of **successor(*tree*, *i*)**
7. The String “**delt**” followed by an integer *i*: print the output of **delete(*tree*, *element*)**
8. The String “**inor**” : print the output of **inorder(*tree*)**
9. The String “**prer**” : print the output of **preorder(*tree*)**
10. The String “**post**” : print the output of **postorder(*tree*)**

Output format

The output (if any) of each command should be printed to terminal on a separate line.

Sample Input

```
srch 25
minm
maxm
pred 25
succ 25
insr 25
srch 25
minm
maxm
pred 25
succ 25
insr 13
insr 50
insr 45
insr 55
insr 18
srch 10
inor
prer
post
```

delt 55
delt 13
delt 25
minm
maxm
stop

Sample Output

NOT FOUND
NIL
NIL
NULL
NULL
FOUND
25
25
-1
-1
NOT FOUND
13 18 25 45 50 55
25 13 18 50 45 55
18 13 45 55 50 25
18
50

2. The Parenthesis Representation of a binary tree is recursively defined, as given below.
- The string () represents an empty tree. **That is, a left bracket followed by a single space followed by a right bracket.**
 - The string (**k left-subtree right-subtree**) represents a tree whose root node has key **k**, **left-subtree** is the left subtree of the root node in Parenthesis Representation and **right-subtree** is the right subtree of the root node in Parenthesis Representation. **That is, a left bracket followed by a single space followed by the key value k followed by a single space followed by the parenthesis representation of left-subtree followed by a single space followed by the parenthesis representation of right-subtree followed by a single space and a right bracket.**

Add a function **paren(tree)** to your program for Question 1. The function should take as input a pointer to the root node of a binary search tree **tree** and print the tree in its Parenthesis Representation.

Input/Output Format (in addition to input/output specification for Question 1)

If the input contains the string “**prep**”, print the BST in its Parenthesis Representation.

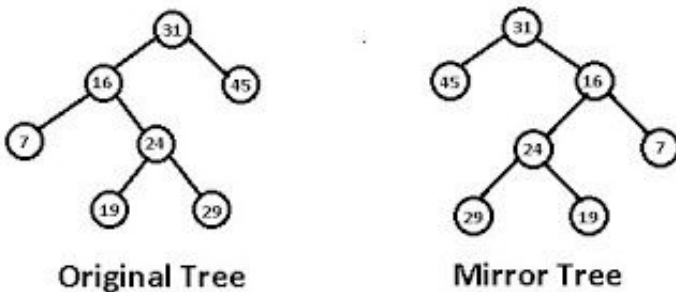
Sample Input

```
insr 43
insr 15
insr 8
insr 30
insr 20
insr 35
insr 60
insr 50
insr 82
insr 70
prep
stop
```

Sample Output

```
( 43 ( 15 ( 8 ( ) ( ) ) ( 30 ( 20 ( ) ( ) ) ( 35 ( ) ( ) ) ) ) (
60 ( 50 ( ) ( ) ) ( 82 ( 70 ( ) ( ) ) ( ) ) ) )
```

- Write a program to construct the mirror of a Binary Search Tree (BST) from the given input recursively. Refer to the figure below for an example of mirror tree.



Add a function **mirror(t)** to your program of qn 2.

mirror(*tree*) - function should take as input a pointer to the root node of a tree **tree** and print its mirror tree in its Parenthesis Representation.

Input Format

Single line containing the parenthesis representation of a BST.

Output Format

Single line containing the parenthesis representation of the mirror BST.

Sample Input:

```
( 31 ( 16 ( 7 ( ) ( ) ) ( 24 ( 19 ( ) ( ) ) ( 29 ( ) ( ) ) ) ) (
45 ( ) ( ) ) )
```

Sample Output

```
( 31 ( 45 ( ) ( ) ) ( 16 ( 24 ( 29 ( ) ( ) ) ( 19 ( ) ( ) ) ) (
7 ( ) ( ) ) ) )
```

4. Given the preorder traversal with unique keys, write a program to construct the Binary Search Tree. Your program should include the following functions.

main() - reads the input as specified in the input format from the terminal and calls the appropriate subfunctions to construct the BST.

Input format

The first line contains a positive integer, representing the number of nodes in the tree.

The second line contains space separated integers, representing a valid preorder traversal of a binary search tree.

Output format

Single line representing the binary tree in the Parenthesis Representation

Sample Input

10
43 15 8 30 20 35 60 50 82 70

Sample Output

(43 (15 (8 () ()) (30 (20 () ()) (35 () ()))) (60 (50 () ()) (82 (70 () ()) ())))

5. Write a recursive program to find the height and the longest path of a binary tree in linear time. Your program should include the following functions.

main() – reads the input as specified in the input format from the terminal and calls the appropriate subfunctions to find the height and the longest path.

height(*tree*) – returns the height of a binary tree specified by *tree*, which is the number of edges between the tree's root and its farthest leaf. Assume that a binary tree with a single node has height zero.

longest_path(*tree*) – return the length of the longest path, which is the maximum number of edges between any two nodes in the tree specified by *tree*.

Input format

Single line representing the binary tree in the parenthesis format.

Output format

Two integers separated by a space denoting height and longest path respectively.

Sample Input

(43 (15 (8 () ()) (30 (20 () ()) (35 () ()))) (60 (50 () ()) (82 (70 () ()) ())))

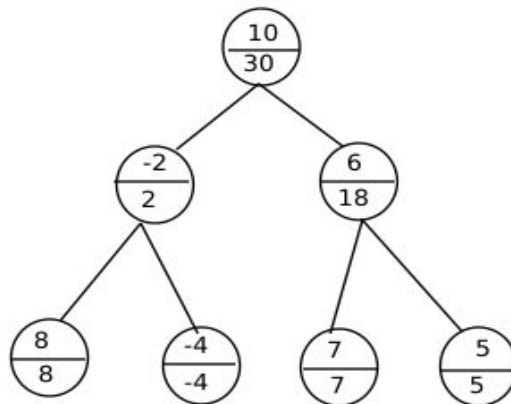
Sample Output

3 6

6. Given a Binary Tree with each of the nodes having positive and negative values. Each node in the tree has an extra field with the sum of its left subtree, right subtree and the node's value itself. Your program should include the following function:

main() – reads the input as specified in the input format from the terminal and calls the appropriate function to find the new tree as specified.

mod_tree(tree) – prints the tree in the specified format from the given binary tree specified by *tree*.



Input format

Single line representing the binary tree in the parenthesis format.

Output format

Single line representing the sum calculated for each node of binary tree in the parenthesis format.

Sample Input

```
( 10 ( -2 ( 8 ( ) ( ) ) ( -4 ( ) ( ) ) ) ( 6 ( 7 ( ) ( ) ) ( 5 ( ) ( ) ) ) )
```

Sample Output

```
( 30 ( 2 ( 8 ( ) ( ) ) ( -4 ( ) ( ) ) ) ( 18 ( 7 ( ) ( ) ) ( 5 ( ) ( ) ) ) )
```

7. Given a binary tree, find the size (number of nodes in the tree) of the largest subtree which is also a Binary Search Tree (BST). Your program should run in Linear time and include the following functions.

main() - reads the input as specified in the input format from the terminal and calls the appropriate function to find the largest BST present in the given binary tree.

maxSubBST(*tree*) - returns the size of the largest BST in the binary tree specified by **tree**.

For Example

If the binary tree is as shown below Fig 1. Identify maximum size BST subtree as shown in Fig 2 and output the number of nodes in Fig 2 as 5.

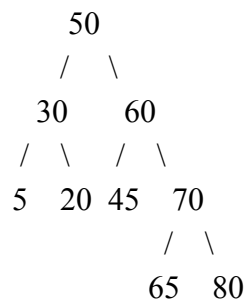


Fig 1

The following subtree is the maximum size BST subtree

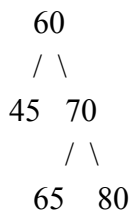


Fig 2

Input format

Single line representing the binary tree in the parenthesis format.

Output format

Single integer representing the size (number of nodes in the tree) of the largest BST in the given binary tree.

Sample Input

```
( 10 ( 40 ( 20 ( ) ( ) ) ( 60 ( ) ( ) ) ) ( 30 ( 90 ( ) ( ) ) ( ) ) )
```