Tree Traversal and BST:

1. Write function to create a Binary Search tree (BST) from a given sequence of integers. Write function to delete a number from a given BST.

2. To binary tree are similar if they are both empty or both nonempty and both have similar left and right subtrees. Write a function to decide whether two binary tree are similar. What is the running time of your program?

3. Construct the unique binary tree from a given in-order and pre-order traversals of the tree.

4. The oldest computer of your lab uses an operating system called "Strange Operating System (SOS)". It has some weird directory structure explained below. You are to write a program to implement the properties of this operating system explained below.

   The initial directory or folder in SOS is called as "root". It will be available when you install SOS. It cannot be deleted. All the other folders/subfolders of your computer will be within root.

   The strange rule about the directory structure of SOS is that each folder can contain at most one folder inside it. This rule applies for root also. However, to make life easier, each folder can contain zero or more files.

   Now, SOS allows following operations on its directory structure:

   a) NEW <FOLDER>: It creates a new folder named <FOLDER> in the current folder <CURR-FOLDER> and takes you inside <FOLDER>. After this operation, you should print: "Made <FOLDER> in <CURR-FOLDER>"

   b) BACK: This operation leads you to the parent folder of current folder <CURR-FOLDER>. Parent folder is the folder in which <CURR-FOLDER> resides. However, this strange operation deletes the <CURR-FOLDER> and all its files and subfolder. So, you should ignore this operation on root with an error message: "Cannot back from root". If this operation is successful, you should print: "Back from <CURR-FOLDER>"

   c) CREATE <FILE>: This operation creates a file named <FILE> in the current folder <CURR-FOLDER>. This newly created file will be the newest file in <CURR-FOLDER>. After this operation, you should print: "Created <FILE> in <CURR-FOLDER>"

   d) DELETE: This operation deletes the oldest file in the current folder <CURR-FOLDER>. You can keep track of age of a file in a folder from when it is created,

by giving it some rank or index. If <CURR-FOLDER> has no file, you should ignore this operation with an error message: “Cannot delete from <CURR-FOLDER>”. If this operation is successful, you should print: “Deleted <FILE> from <CURR-FOLDER>”, where <FILE> is the name of file which is deleted.

Input:

Each line of input will be one of the four operations of SOS explained above. -1 denotes end of input. <FOLDER> and <FILE> will be alphanumeric strings of length at most 20.

CREATE f: Initially there will only be “root” folder. So, file f is created in root.

NEW ab: folder “ab” will be created in “root” folder. Now you will be inside folder “ab”. So, subsequent operation will be inside folder “ab”

CREATE f1: file f1 will be created inside folder “ab”

CREATE f2: file f2 will be created inside folder “ab”.

DELETE: oldest file inside folder “ab” will be deleted,i.e. file f1

BACK: you will return to parent folder of “ab” while deleting folder “ab”,all its subfolders & file

Output:

Print corresponding message to each operation, in a new line.

Test1:

CREATE f1

NEW abc

NEW def

CREATE f2

CREATE f3

DELETE

BACK

CREATE f4

DELETE

BACK

DELETE

DELETE

BACK

NEW ghi

-1

Output:

Created f1 in root

Made abc in root

Made def in abc

Created f2 in def

Created f3 in def

Deleted f2 from def

Back from def

Created f4 in abc

Deleted f4 from abc

Back from abc

Deleted f1 from root

Cannot delete from root

Cannot back from root

Made ghi in root