Indian Institute of technology, Guwahati
Department of Computer Science and Engineering
Data Structure Lab: (CS210)

Offline Assignment: 2

Date: 14th August, 2016. **Total Marks: 20**
Deadline: 10PM, 20th August, 2017. (Hard Deadline)

1. **[SOS]** 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. **[10]**

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

Test2:
NEW assignment
NEW o1
CREATE problem1
DELETE
CREATE problem2
CREATE problem3
BACK
NEW o2
CREATE p1
CREATE p2
CREATE p3
DELETE
NEW o21
-1

Output:
Made assignment in root
Made o1 in assignment
Created problem1 in o1
Deleted problem1 from o1
Created problem2 in o1
Created problem3 in o1
Back from o1
Made o2 in assignment
Created p1 in o2
Created p2 in o2
Created p3 in o2
Deleted p1 from o2
Made o21 in o2

2. **[Huffman Encoding]** You are given a character string. First, you need to find to the frequency of each character in the string. Write a program to create **"Huffman Tree"** using character frequencies and find the prefix code for each character. When two least frequent nodes are merged together to form an internal node, the node having lesser frequency should be made left child. In case of tie, the node which comes lexicographically first will become left child. Least frequent node is node having least frequency. If there are more than two nodes having least frequency, two nodes which come first in lexicographic order will be considered.

   Input: Character String
   Output: Prefix code for each character **[10]**

   Suppose input string is **"aaabbcde".** Frequency of a, b, c, d & e are respectively 3,2,1,1 & 1. Since all three of c, d & e are having frequency 1. But c comes lexicographically prior than d and e. So first least frequent node will be c and second least frequent node will be d. e will be the third one. So, first c and d will be merged together to form internal node. c and d will be left and right child respectively of cd (newly created node). Now, nodes available for merging are a, b, cd, e. Now, e and b will be merged making new node having frequency 3. Now nodes for merging are a, cd, eb. Now cd and a will be merged forming cda. Now eb and cda will be merged.

   Prefix codes for characters are: a = 11 b = 01 c = 100 d =101 e = 00

   Test1:
   Input:
   huffman

   Output:
   h=011 u=00 f=10 m=110 a=010 n=111

   Test2:
   Input:
   aaaaaabbccddeefffff

   Output:
   a=11 b=000 c=001 d=010 e=011 f=10