DFS LAB – ASSIGNMENT 4

MTech(CS) I year 2019–2020

Deadline: 15 December, 2019

Total: 60 marks

SUBMISSION INSTRUCTIONS

1. Naming convention for your programs: `cs19xx-assign3-progy.c`

   IMPORTANT: Insert a single alpha-numeric string of your choice, 6-8 characters long, in the name given above as shown in the examples below. Think of this string as something like a security password, except that you are not required to remember the string. Examples: `cs1940-assign3-x19jdh4-prog1.c`, `ppo03wws-cs1940-assign3-prog2.c`, `cs1940-assign3-prog2-jsiwm7de.c`

2. To submit a file, go to the directory containing the file and run the following command from your account on the server (IP address: 192.168.64.35)

   ```
   cp -p name-of-your-file ~dfslab/2019/assign4/cs19xx/
   ```

   If you want to submit your files from a computer with a different IP address, run

   ```
   scp -p name-of-your-file \
   mtc19xx@www.isical.ac.in:/user1/perm/pdslab/2019/assign4/cs19xx/
   ```
   (enter your password when prompted).

   To submit all `.c` and `.h` files at one go, use
   `cp -p *.c *.h ~dfslab/2019/assign4/cs19xx/` or similar.

NOTE: Unless otherwise specified, all programs should take the required input from stdin, and print the desired output to stdout.

Q1. Data transformation is often necessary as a pre-processing step for converting data from one format to another, typically from the format of a source system into the required format of a destination system. There are standard data transformation tools that can convert one type of delimited file into another type of delimited file. As for example, converting CSV files to XLS files is nothing but converting comma delimited files to tab delimited files. Note that, the common delimiters that are used in practice are tab (fixed number of spaces), comma (,), semicolon (;), pipe (|), quotes (‘, ’, “, ”), braces ({, }), or slashes (/, \).

Consider a scenario where you are given a noisy file with mixture of delimiters. You have to recognize the most frequent delimiter (say ⌈) in the file and it has to be converted to a format that contains only ⌈ as the delimiter. Write a program to perform the said task. [20 marks]

Input Format

Input will be provided in a separate file (name to be taken from command-line arguments) in the following format. The input will contain delimited text with alphanumeric or special characters. The contents of the input file may spread across multiple lines.

Output Format

Output is to be printed on a separate file (name to be taken from command-line arguments). The output will contain consistently delimited text taken from the input file in the exact order. The delimiter must be the most frequently used delimiter in the input text. In case of a tie, print `CONFUSING` on the standard output.

Command-line Arguments

```
./prog1 <input_filename> <output_filename>
```

Sample Input 0

```
Contents of input.txt:
2,3,5,4
3 7,8,4
3,9 10,4
2,2 2,4

Input from command-line:
./prog1 input.txt output.txt
```

Sample Output 0

```
Contents of output.txt:
2,3,5,4
3,7,8,4
3,9,10,4
2,2,2,4
```

Sample Input 1

```
Contents of input.txt:
a/b/c/d
e|f|g|h
i/j/k/l
m/n/o/p
q/r/s/t

Input from command-line:
./prog1 input.txt output.txt
```

Sample Output 1

```
Contents of output.txt:
a/b/c/d
e/f/g/h
i/j/k/l
m/n/o/p
q/r/s/t
```

Sample Input 2

```
Contents of input.txt:
1   2   3
4 5 6
7,8,9

Input from command-line:
./prog1 input.txt output.txt
```

Sample Output 2

```
CONFUSING
```

Q2. Modern smartphones let you search contact details directly from the dialer application (say `DIAL APP`). As soon as you type a single digit and go ahead on the numeric keypad of `DIAL APP`, it predicts the list of numbers (as well as the associated names) you might want to search for. Formally, this mechanism is termed as Predictive Text (also known as `T9`) technology.

Given a set a contact details (contact names along with numbers), you need to store them in a way such that they can be searched efficiently using `T9` technology for an input numeric string. Note that, the search string might be present anywhere within the contact numbers. Write a program that will list up all predicted contact details for the given search string in the alphabetical order of contact names. [20 marks]

Input Format

Input will be provided in two different forms. The contact details will be provided in a separate file (name to be taken from command-line arguments) in the following format.

The first line of input will contain n, where n is the number of contacts. This will be followed by n more lines each containing the contact names and numbers separated by spaces, not necessarily in alphabetical order.

The numeric search string will also be provided as a command-line argument alongside the filename.

Output Format

Output is to be printed on the standard output. Output will print the list of all predicted contact names and numbers, separated by spaces, for the given search string in alphabetical order of contact names.

Command-line Arguments

```
./prog2 <input_filename>
```

Sample Input 0

```
Contents of input.txt:
3
Tamaltaru Pal 9000000002
Mandar Mitra 9000000001
Malay Bhattacharyya 9000000000

Input from command-line:
./prog2 input.txt 900
```

Sample Output 0

```
Output on the terminal:
Malay Bhattacharyya 9000000000
Mandar Mitra 9000000001
Tamaltaru Pal 9000000002
```

Sample Input 1

```
Contents of input.txt:
3
ISI 9876543210
IITKGP 9976543210
IIMCAL 9876543299

Input from command-line:
./prog2 input.txt 99
```

Sample Output 1

```
Output on the terminal:
IIMCAL 9876543299
IITKGP 9976543210
```

Sample Input 2

```
Contents of input.txt:
4
DFS1 8888888888
DFS2 7777777777
DFS3 6666666666
DFS4 9999999999

Input from command-line:
./prog2 input.txt 6789
```

Sample Output 2

```
Output on the terminal:
```

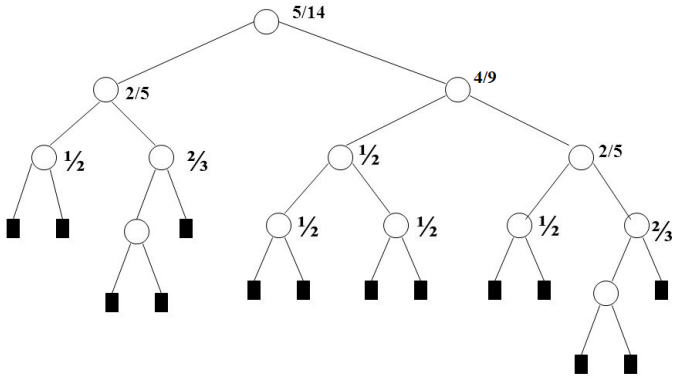
Q3. To obtain a logarithmic search time it is required to bound the height of a BST. A common way to do this is to balance the heights of the two subtrees for each node in the BST. The AVL tree, Red-Black tree are such popular schemes. Now consider another kind of balancing mechanism where we balance the weights (instead of height) of the two subtrees for each node. We define weight-balanced tree (say WB tree) as a binary search tree that stores the sizes of subtrees in the nodes.

The idea of this WB tree was first proposed in the following paper:

J. Nievergelt and E. M. Reingold, Binary Search Trees of Bounded Balance, *SIAM Journal on Computing*, 2(1):33-43, 1973. (Link: `https://epubs.siam.org/doi/pdf/10.1137/0202005`)

The size of an internal node (say n) is the sum of sizes of its two children plus one, i.e. $size(n) = size(n.left) + size(n.right) + 1$. Based on the size, the weight is defined as $weight(n) = size(n) + 1$. By definition, the size of a leaf node is zero. A WB tree is said to be of bounded balance α if $weight(n.left) \geq \alpha weight(n)$ and $weight(n.right) \geq \alpha weight(n)$ (see the figure follows).

Your task is to implement the WB tree and study its performance. For a given input sequence of n numbers organized as an arbitrary binary tree, count the total number of rotations required and final height obtained to convert it to a WB tree. Operations that convert the tree must ensure that the weight of the left and right subtrees of every node remain within some factor α of each other, using the same re-balancing operations used for AVL trees, namely rotations and double rotations. Note that, a completely balanced tree has $\alpha = \frac{1}{2}$. [20 marks]



Input Format Input will be provided via standard input in the following format. The first line of input is the balance factor α. The second line of input will contain n, denoting the number nodes in the binary tree. This will be followed by n more lines. Each of these remaining lines will correspond to one node in the tree and will consist of 3 integers: the data (an integer to be stored in the node), the line number (count starts from 1 representing the line where the root appears) of the node corresponding to the left child (-1 if there is no left child), and the line number corresponding to the right child (-1 if there is no right child).

Output Format The output will print the total number of rotations required and final height obtained separated by spaces to derive the required WB tree.

Sample Input 0

```
1/2
3
10 -1 2  <-- line number 1 starts from here
20 -1 3
30 -1 -1
```

Sample Output 0

```
2 2
```