Indian Statistical Institute

Semester-I 2019-2020

M.Tech.(CS) - First Year

Lab Test 3 (22 October, 2019)

Subject: Data and File Structures Laboratory

Total: 60 marks Duration: 4 hrs.

SUBMISSION INSTRUCTIONS

1. Naming convention for your programs: `cs19XX-test3-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: `cs19XX-test3-abcdef-prog1.c`, `mnopqr-cs19XX-test3-prog2.c`, `cs19XX-test3-prog2-uvwxyz.c`
2. When you have finished, copy all your files to `~dfslab/2019/labtest3/cs19XX/` on 192.168.64.35 using
   `scp <file names> cs19XX@192.168.64.35:~dfslab/2019/labtest3/cs19XX/`

1. **(20 marks)** French flags consist of three bands of blue, white and red as shown in the figure below. The width and height of each band are w and h units, respectively. Suppose you are given a collection of blue, white and red bands (each $w \times h$ units in size) in some arbitrary order. Your job is to stitch these bands together to form the largest French flag possible. Note that, the flag that you are able to make should have the same aspect ratio (height-to-width proportion) as the official flag.

   Write a program that can efficiently perform the said task by making only a single pass over the bands. For the sake of simplicity, consider that the official French flag has an aspect ratio of 1:3.

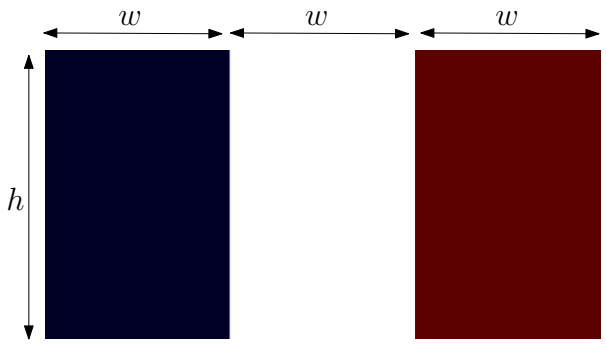

Figure 1: The French flag.

Input Format

Input will be provided via the standard input in the following format. The first line of input consists of the number of colored bands available to you. The next line consists of the characters 'B', 'R' and 'W' (corresponding to the colors Blue, Red and and White, respectively) in some arbitrary order, separated by spaces.

Output Format

The output should show the layout in which the bands should be stitched in order to form the biggest flag possible. Please see the examples below for details. If no flag can be designed from the given bands, output should print nothing.

Sample Input 0

```
9
W R B W R B W R B
```

Sample Output 0

```
B W R
```

Sample Input 1

```
15
W R B W R B W R B R R B B W W
```

Sample Output 1

```
B B W W R R
B B W W R R
```

Sample Input 2

```
42
W W R R B W W R W R R W B W W R B W B R B R B W B R B B R R W R R W W W R W W W W R
```

Sample Output 2

```
B B B W W W R R R
B B B W W W R R R
B B B W W W R R R
```

2. **(20 marks)** Recall that a max heap is a complete binary tree that satisfies the max heap ordering property, i.e., each node in the tree has a key which is less than or equal to the key of its parent. Suppose a *Search Max Heap* is a special kind of max heap in which all the values in the left subtree of any arbitrary node are less than all the values in the right subtree of the node.

   Given a complete binary search tree (BST), convert it into an equivalent *Search Max Heap* involving minimal operations.

   Input Format

Your program should take a single filename as its command-line argument. The given complete BST will be stored in this file in the following format. The first line will contain n, the number of nodes in the BST. This will be followed by n more lines. Each of these remaining lines will correspond to one node in the tree (with the first line representing the root node), and will consist of 3 integers: the data (an integer to be stored in the node), the line number 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

Output is to be printed on the standard output in the following format. The output should contain n lines. Each of these lines will correspond to one node in the heap (with the first line representing the root node), and will consist of 3 integers, as described above. Assume that the first line (corresponding to the root) is numbered 0.

Sample Input 0

```
3
2 1 2
1 -1 -1
3 -1 -1
```

Sample Output 0

```
3 1 2
1 -1 -1
2 -1 -1
```

Sample Input 1

```
7
8 1 2
6 3 4
10 5 6
5 -1 -1
7 -1 -1
9 -1 -1
11 -1 -1
```

Sample Output 1

```
11 1 2
7 3 4
10 5 6
```

```
5 -1 -1
6 -1 -1
8 -1 -1
9 -1 -1
```

3. **(20 marks)** The quality of research publications of a scientist is often characterized by his citation metrics. Every research publication has a citation count that represents the number of times it has been referred in the other research publications, thereby indicating its credit.

   The h-index is one of the popular citation metrics that is defined as the `maximum` value of h such that the given scientists has published h papers that have each been cited at least h number of times. Suppose you are given with the citation counts of a scientist by his research publications in the chronological order (following the order in which they were published). Write a program to efficiently compute the h-index of the scientist.

   Input Format

   Input will be provided as command-line arguments as a series of numbers representing the citation counts of research publications in the chronological order.

   Output Format

   Output is the h-index that is to be printed on the standard output.

   Sample Input 0

   ```
   10 2 1 5 11 4 3
   ```

   Sample Output 0

   ```
   4
   ```

   Sample Input 1

   ```
   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   ```

   Sample Output 1

   ```
   1
   ```

   Sample Input 2

   ```
   201 305
   ```

   Sample Output 2

   ```
   2
   ```