Contest	[Fall17][B1] Assignment #3
Problem	AVL Tree (AVLT)
Input file(s)	`input.txt`
Output file	`output.txt`
Time limit	10 sec.
Memory limit	128 MB

Statement

`Part A:` you will implement `Your Own AVL Tree` which will extend your `Binary Search Tree (BST) Class` from the previous question, and `Use Trinode Restructuring` to ensure that the tree stays balanced after `insertions and deletions`.

`Part B:` Counting the total number of smallerelements than each element in a given unsortedsequence using AVL Tree. In this case, given a sequence of integers, we are interested in finding the sum of the number of smaller elements than each element in the given sequence. For example, let 2 4 1 8 be the given unsorted sequence. Then, For 2 there is only one element smaller than it, i.e., 1 `(1)`

For 4, it is 2 and 1 `(2)`

For 1, there is none `(0)`

For 8, it is 4, 2, and 1 `(3)`

Our desired number is:`1 + 2 + 0 + 3 = 6`

`What you are required to do is as follows:`

1. Given a sequence of integers (input.txt), you will create an AVL Tree using your implementation.

2. Use the resulting AVL Tree to find this sum and print it (output.txt)

Example

`input.txt`

```
2 4 1 8
```

`output.txt`

```
6
```

Contest	[Fall17][B1] Assignment #3
Problem	Binary Search Tree
Input file(s)	`input.txt`
Output file	`output.txt`
Time limit	10 sec.
Memory limit	128 MB

Statement

In this question, you will implement `Your Own Binary Search Tree` as stated in `Part A, and then prints its mirrored version as stated in Part B:`

`Part A:` First you are required to implement `Your OWN Binary Search Tree (BST) Class` as an abstract data type. This means that you will first create a Java Interface and then create the above class that implements the interface. Your implementation should support the following methods:

`find(k):` returns all entries with key k if they exist, null otherwise.

`insert(k):` inserts an entry with key k.

`remove(k):` removes all entries with key k.

`traverse():` gets the string of the inorder traversal of the tree.

`print():` prints out the tree.

`Part B:` Next, you will implement the method `mirror(),` which prints a mirrored version of the original BST, called BSMT, with left and right children of all non-leaf nodes interchanged.

`NOTE:`

Your `Input.txt` contains the

1. Sequence to build BST,

2. The value you want to find

3. The value you want to remove

4. The value you want to insert.

You final code>Output.txt consists of multiple lines and items:

1. The first line contains the found value,

2. Second line contains the inorder traversal,

3. At third BST,

4. Finally, at fourth BSMT.

Example

`input.txt`

```
11 6 8 19 4 10 17 43 49 31 (Input sequence)
19 (Find)
49 (Remove)
44 (Insert)
```

---

output.txt

```
19
4 6 8 10 11 17 19 31 43 44 (Inorder)
BST:
11 6 19
6 4 8
19 17 43
8 10
43 31 44
BSMT:
11 19 6
19 43 17
6 8 4
43 44 31
8 10
```

Please note that

```
in BST (at third) and BSMT (at fourth) each line shows the parent node and its chi
ldren from left to right.


The first line is for the root node and its children, the subsequent lines show th
e same relationship for root’s left and right child,


respectively, and the pattern continues for their descendants.
```

Contest	[Fall17][B1] Assignment #3
Problem	B-Tree

Input file(s)	`input.txt`
Output file	`output.txt`
Time limit	10 sec.
Memory limit	128 MB

Statement

`Part A:` You will implement `Your Own B-Tree of order 3 (i.e. m = 3) which Includ Your Own “Insertion” and “Find” operations,` and `Use Trinode Restructuring (Your Own)` to ensure that the tree stays balanced after every insertion.

`Part B:` Use `Your Own B-Tree implementation` and then `Inorder traversal method (i.e. Tree Balancing Property)` to find the `Sorted sequence (for both Number and Alphabets) of a given unsorted sequence`. See the Examples for a better understanding.

`Note:` You must need to satisfy the B-Tree properties. Violation of any of property/ies will set your grade to zero for this particular question.

Example 1

`input.txt`

```
5 16 22 45 2 10 18 30 50 12 31
```

`output.txt`

```
2 5 10 12 16 18 22 30 31 45 50
```

Example 2

---

input.txt

```
M D G Q T A E C F K H L P N W R X S Z Y
```

---

output.txt

```
A C D E F G H K L M N P Q R S T W X Y Z
```