Md. Saif Niaz
ID# 100555440

Assignment 3

CSCI 2010U - Principles of Computer Science

Due: 27 November 2015 at Midnight

Introduction

This assignment focuses on constructing, balancing, and searching in binary trees. In terms of algorithms you are asked to implement AVL algorithm for balancing your binary trees.

Stream Processing

You are provided a Stream class that generates random integers on demand. The Stream class is implemented in Stream.java file. You are also provided a StreamProcessor class that consumes (stores) the integers generated by the Stream class. The length of the stream is not known *a priori*, so the StreamProcessor class uses a sorted binary tree to stores the incoming integers. Classes BNode and BTree implement the sorted binary tree used here.

Searching

At any time during the process the user can issue a search queries of the following form to the stream processor:

- search(value); and
- at(index).

The first query returns the index(s) at which the value appeared in the stream (assuming that the first integer from the stream is at index 0) or -1 if the value never appeared in the stream. While the second query returns the value that appeared at a particular index in the stream.

Example

Stream: 4 5 3 4 6 6 2 10 10 100 29 …

search(5) returns 1
search(4) returns [0, 3]
search(99) returns -1
at(4) returns 6

Efficient Data Structures

A naïve sorted binary tree implementation might work for implementing the **search** query, but it will perform miserably with the **at** query. So you need to implement the AVL algorithm to balance your binary tree in order to handle both queries efficiently.

Implementation Goals

- Complete the binary tree code to satisfy **search** and **at** queries. *Hint: you need to store two binary trees, one for each query.*
- Implement AVL tree balancing algorithm.
- You will need another dynamic structure to hold the command line parameters, such as search queries.

Evaluation

Write two programs bt and btavl that will be used to time your program. We will run your programs as follows

```
java  bt <seed> <integer range> <stream length> <search=val1> <search=val2> … <search=valn> <at=ind1>
<at=ind2> … <at=indn>
```

and

```
java  btavl <seed> <integer range> <stream length> <search=val1> <search=val2> … <search=valn> <at=ind1>
<at=ind2> … <at=indn>
```

The description of the command line parameters is below:

- seed: the seed for the random number generator, it allows us to generate the same stream over and over again (good for testing purpose)
- integer range: the maximum integer that may appear in the stream
- stream length: the length of the stream
- search=val: issue search(val) query
- at=ind: issue at(ind) query

The output of the program must strictly adhere to the following format:

```
search(val1) = [one or more indices or -1]
search(val2) = [one or more indices or -1]
…
search(valn) = [one or more indices or -1]
at(ind1) = value
at(ind2) = value
…
at(indn) = value
time (tree setup) = x ms
time (average search) = x ms
time (average at) = x ms
```

Carry out the necessary tests to complete the following table

	AVL Tree	AVL Search	AVL At	Tree	Search	At
100	7.0ms	0.67 ms	0.33 ms	6.0 ms	0.67 ms	0.67 ms
1000	24 ms	1.67 ms	2.33 ms	21.0 ms	2.00 ms	2.00 ms
50000	163 ms	10.67 ms	12.0 ms	128 ms	12.00 ms	11.67 ms
500000	2122 ms	43.33 ms	44.67 ms	2023 ms	40.00 ms	39.67 ms
1000000	36634 ms	1299.3 ms	1244.67 ms	34044 ms	855.67 ms	886.67 ms

Submission

Please follow these instructions to the letter. Thank you.

Create a zip file called A3_<studentnumber>.zip with at least the following three files: bt.java, btavl.java and a3.pdf. All files should contain your name and student number at the top.