# Homework 6
# Trees and heaps

CS 5060 Intensive Programming, Fall 2012

100 points

Due: 3:59 pm October 31, 2012

## Binary search tree and binary heap (40 points).

Implement a binary search tree and an array based binary heap. Create your own `BSTree` interface and make sure your `MyBSTree` class implements it. Also, create your own `BHeap` interface and make sure your `MyBHeap` class implements it. Make sure you reuse the tree and heap code for other problems, instead of copying it. ***Points will be deducted if you copy the code for each problem.***

Note: All input must be read from the standard input stream, and all output must be written to the standard output stream. For this assignment, assume the input is correct.

## Problem 0a: Binary search tree (20 points)

Implement a binary search tree.

**Input:** The input begins with the number $t$ of test cases in a single line ($t \leq 100$). Each of the next $t$ lines starts with a number $n$ ($1 \leq n \leq 1000$) followed by a list of $n$ operations. There are two types of operations: insert and remove. An insert operation is represented by a character 'i' followed by a number $m$ ($1 \leq m \leq 1000000$) to add to the tree (***ignore repeated numbers***). A remove operation is represented by a character 'r' followed by a number $m$ ($1 \leq m \leq 1000000$) to remove from the tree.

**Output:** For each test case output the contents of the tree, in pre-order, after all the operations have been performed. Consecutive elements should be separated by a single space.

**Example:**

**Input:**

```
4
4 i 5 i 0 i 6 i 0
5 i 5 i 4 i 3 i 2 i 1
6 i 5 i 3 i 7 i 6 i 4 i 2
7 i 5 i 3 i 7 i 6 i 4 i 2 r 3
```

**Output:**

```
5 0 6
5 4 3 2 1
5 3 2 4 7 6
5 4 2 7 6
```

## Problem 0b: Binary heap (20 points)

Implement an array based binary heap.

**Input:** The input begins with the number $t$ of test cases in a single line ($t \leq 100$). Each of the next $t$ lines starts with a number $n$ ($1 \leq n \leq 1000$) followed by a list of $n$ operations. There are two types of operations: insert and remove minimum. An insert operation is represented by a character 'i' followed by a number $m$ ($1 \leq m \leq 1000000$) to add to the heap (***allow repeated numbers***). A remove minimum operation is represented by a character 'r'.

**Output:** For each test case output the contents of the array after all the operations have been performed. Consecutive elements should be separated by a single space.

**Example:**

**Input:**

```
4
4 i 5 i 0 i 6 i 0
5 i 5 i 4 i 3 i 2 i 1
6 i 5 i 3 i 7 i 6 i 4 i 2
7 i 5 i 3 i 7 i 6 i 4 i 2 r
```

**Output:**

```
0 0 6 5
1 2 4 5 3
2 4 3 6 5 7
3 4 7 6 5
```

**Grading:** For each data structure, the interface is worth 5 points, the implementation is worth 10 points, and the test program is worth 5 points.

For each data structure, make sure you use the interface name when declaring instances (not the class name). For example, you should write:

```
BSTree tree = new MyBSTree();
BHeap heap = new MyBHeap();
```

This is worth 2 points for each problem in the assignment.

## Solve the following problems (30 points).

Use the binary tree implemented in problem 0.

Note: All input must be read from the standard input stream, and all output must be written to the standard output stream. For this assignment, assume the input is correct.

## Problem 1: Sum of depths (15 points)

**Input:** Same as problem 0a.

**Output:** For each test case output the sum of the depths of all the nodes.

**Example:**

**Input:**

```
4
4 i 5 i 0 i 6 i 0
5 i 5 i 4 i 3 i 2 i 1
6 i 5 i 3 i 7 i 6 i 4 i 2
7 i 5 i 3 i 7 i 6 i 4 i 2 r 3
```

**Output:**

```
2
10
8
6
```

## Problem 2: Balanced tree (15 points)

A binary tree is balanced if the height of the two sub-trees of every node differ by at most 1. An equivalent recursive definition is:

- the difference between the heights of the left sub-tree and the right sub-tree is at most 1,
- both sub-trees are balanced.

**Input:** Same as problem 0a.

**Output:** For each test case determine if the tree is balanced.

**Example:**

**Input:**

```
4
4 i 5 i 0 i 6 i 0
5 i 5 i 4 i 3 i 2 i 1
6 i 5 i 3 i 7 i 6 i 4 i 2
7 i 5 i 3 i 7 i 6 i 4 i 2 r 3
```

**Output:**

```
YES
NO
YES
YES
```

## Sorting problems (30 points).

## Problem 3: Tree sort (15 points)

Use a binary search tree to sort a list of numbers. What is the running time of your algorithm?

**Input:** The input begins with the number $t$ of test cases in a single line ($t \leq 100$). Each of the next $t$ lines starts with a number $n$ ($1 \leq n \leq 1000$) followed by a list of $n$ numbers $m$ ($1 \leq m \leq 1000000$) to sort.

**Output:** For each test case output the numbers in increasing order.

**Example:**

**Input:**

```
2
4 5 0 6 0
5 5 4 3 2 1
```

**Output:**

```
0 0 5 6
1 2 3 4 5
```

## Problem 4: Heap sort (15 points)

Use a binary heap to sort a list of numbers. What is the running time of your algorithm?

**Input:** The input begins with the number $t$ of test cases in a single line ($t \leq 100$). Each of the next $t$ lines starts with a number $n$ ($1 \leq n \leq 1000$) followed by a list of $n$ numbers $m$ ($1 \leq m \leq 1000000$) to sort.

**Output:** For each test case output the numbers in increasing order.

**Example:**

**Input:**

```
2
4 5 0 6 0
5 5 4 3 2 1
```

**Output:**

```
0 0 5 6
1 2 3 4 5
```

## Submission.

Submit a `zip` file with the following files:

1. Code files `BSTree.java`, `MyBSTree.java`, `BHeap.java`, and `MyBHeap.java` with your interfaces and implementations of the data structures; and code files `MyBSTreeTest.java` and `MyBHeapTest.java` with the solution to problems 0a and 0b.
2. Four code files `SumOfDepths.java`,`BalancedTree.java`,`TreeSort.java` and `HeapSort.java` with the solutions to problems 1, 2, 3,and 4, respectively.

Include your name and A number at the top of each source file. Name the `zip` file `hw06_firstName_lastName.zip`. For example, if your name is John Smith, name the file `hw06_John_Smith.zip`.