Homework 3: Binary Trees

Handed out: **October 23, 2012**
Due: **October 30, 2012**

Submission

Submit your answers to the questions below **as a PDF document**. Failure to do so will result in points loss. If you do not know how to generate a PDF document, do ask! You will use turnin for this submission and name your PDF file <your_first_name>_<your_last_name>.pdf, as you did for the first project. The turnin command will therefore be:

```
% turnin -c cs251 -p homework3 <your_first_name>_<your_last_name>.pdf
```

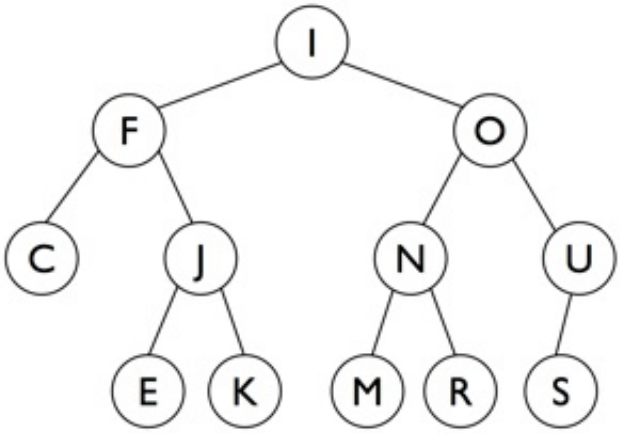
Grading

There are a total of 100 points for this homework, distributed as follows.

Question 1. (15 points)

For the following binary tree, give the

i) Inorder traversal
ii) Preorder traversal
iii) Postorder traversal



Question 2. (20 points)

I) Draw the MinPQ (implemented as a heap) using the following letters (entered in the given order)

II) Show them in the order stored in an array representation

a) Q, F, Y, I, O, R, A, N, H, Z, P, X

b) Remove the MIN 3 times

Question 3. (25 points)

Implement an Iterator that iterates only over the leaves (external nodes) of a binary tree and implements the following API:

```
boolean hasNext();
Object next();
void remove();
```

Question 4. (15 points)

Match up each worst-case quantity on the left with the best matching order-of-growth term on the right. You may use a letter more than once.

___ Height of a binary heap with N keys.	*A) 1.*
___ Height of a BST with N keys.	*B) log N*
___ Number of comparisons to sort N equal keys using our standard version of quicksort.	*C) N*
___ Number of comparisons to sort N equal keys using 3-way quicksort.	*D) N log N*
___ Time to iterate over the keys in a BST using inorder traversal.	*E) N^2*
___ Number of array access to insert a key into a BinarySearchST of size N.	*F) 2^N*

Question 5. (25 points)

```
public Key mystery(Key key) {
    Node best = mystery(root, key, null); if (best == null) return null;
    return best.key;
}

private Node mystery(Node x, Key key, Node best) {
    if (x == null) return best;
    int cmp = key.compareTo(x.key);
    if (cmp < 0) return mystery(x.left, key, x);
    else if (cmp > 0) return mystery(x.right, key, best);
    else return x;
}
```

Consider the binary search tree method above. What does mystery(key) return? Circle the best answer.

1. ***Floor***: the largest key in the symbol table ≤ the search key.
2. ***Ceiling***: the smallest key in the symbol table ≥ the search key.
3. ***Get***: the key in the symbol table equal to the search key if it's there, otherwise null.
4. ***Bad code***: null pointer exception or infinite loop on some inputs.