Homework Written #2:
Solve the following 3 problems. For each problem your solution will include a written portion with analysis, followed by a C++ implementation.

1. Given a sorted array of *n* comparable items *A*, and a search value *key*, return the position (array index) of *key* in *A* if it is present, or -1 if it is not present. If *key* is present in *A*, your algorithm must run in order O(log *k*) time, where *k* is the location of *key* in *A*. Otherwise, if *key* is not present, your algorithm must run in O(log *n*) time.

   Turn in:

   a. A written description of your algorithm, along with an explanation for why it works, and an analysis of your run time.
   b. A C++ implementation of your solution. Use: 'int fastFind(vector<double> &A, double key);' as your function prototype.

2. Given a sorted array of *n* comparable items *A*, create a binary search tree from the items in *A* which has height $h <= \log_2 n$. Your algorithm must create the tree in O(*n*) time.

   Turn in:

   a. A written description of your algorithm, along with an explanation for why it works, and an analysis of your run time.
   b. A C++ implementation of your solution. Use: 'node* buildTree(vector<double> &A, int start, int end);' as your function prototype, which will create a tree from the items in A from indices start to end (inclusive), and return a pointer to the root node of the resultant tree.

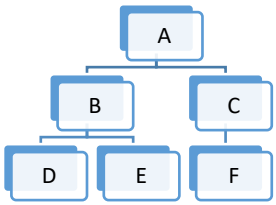
3. Traversing the tree level by level: For the following question, assume binary trees consist of nodes from the following class:

```
class node{

public:
      int data;
      node * left;
      node * right;
};
```

Write a method 'void levelOrderTraversal(node * r)' which prints the items of a binary tree rooted at node r in a "level order". That is, the first item printed is the value contained in the root node r, the next items printed are the children of the root, the next items printed are the grandchildren of the root, etc. Your algorithm must run in O(n) time. Hint: You may use the STL queue in your solution.

For example, the following tree would be printed in the order: A B C D E F



Turn in:

a. A written description of your algorithm, along with an explanation for why it works, and an analysis of your run time.
b. A C++ implementation of your solution. Use: 'void levelOrderTraversal(node * r);' as your function prototype.