MA346m Quiz 01 ANSWERS 6/10/17

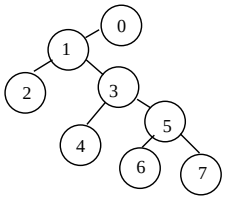
(1). Give a simple example where two different binary trees are the same in preorder and postorder. **Answer:** below.

(2). Write a piece of C code which given a pointer p to a binary tree node, returns the preorder successor of p, as sketched out in lectures.

```
if ( p->left != NULL )
  return p->left;
else if ( p->right != NULL )
  return p->right;
else
{
  BTREE_NODE * q = p->parent;
  while ( q != NULL && p == q->right )
  { p = q; q = p->parent; }

  return q;
}
```

(3). The tree below is labelled according to *preorder*. Write these labels in inorder. This should serve as a guide to the remainder of the question.



`2 1 4 3 6 5 7 0`

Write a piece of C code

```
BTREE_NODE * build ( int i, int j, int a[] )
```

which, given both the inorder and preorder sequences of nodes in a binary tree, reconstructs the tree. The arguments imply that a subtree is stored between indices i and j: the initial call is `build(0, n-1, a)`. To make it easier, you can assume that the preorder ranks are given in an array according to inorder rank. *Recursion* is very useful here.

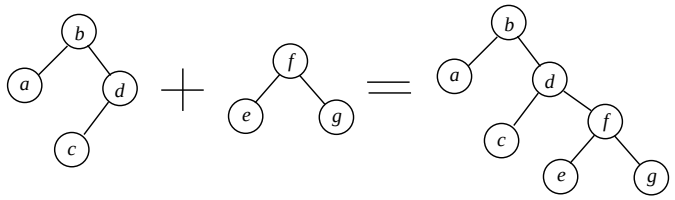
```
BTREE_NODE * build ( int i, int j, int a[] )
{
  if ( i > j )
    return NULL;
  else
  {
    int minpos = i; int k;
    for (k = i+1; k <= j; ++k)
      if ( a[k] < a[minpos] )
        minpos = k;

    BTREE_NODE * q = make_btree_node ( a[minpos] );
    q->left = build ( i, minpos-1, a );
    q->right = build ( minpos+1, j, a );
    return q;
  }
}
```

(4). Write a piece of C code

```
join ( BTREE * t1, BTREE * t2, BTREE * t3 ) ...
```

where `t3` is the new tree. It takes the nodes from the first two trees and modifies the pointers so that the nodes reappear in `t3`, preserving inorder within `t1` and `t2`, and making all nodes from `t2` follow those from `t1` with respect to inorder. This 'destroys the arguments:' `t1` and `t2` will no longer be correct. Example:



Try to make it efficient: $O(h_1 + h_2)$ where h_1, h_2 are the heights of t_1, t_2.

```
if ( t1->root == NULL )
  t3->root = t2->root;
else if ( t2 -> root == NULL )
  t3->root = t1->root;
else
{
  BTREE_NODE * p = t1->root;
  while ( p->right != NULL )
    p = p->right;
  t2->root->parent = p;
  p->right = t2->root;
  t3->root = t1->root;
}
```