* Stacks
* FILO
* Array Implementation: keep int topIndex
* Reversing
* ListNode (listnode, insert in front of the list. O(1) for everything
* Queues
* FIFO
* ListNode with frontPtr, backPtr. Both start at NULL
* Move backptr as pushes come in, front ptr’s are moved when it is popped.
* Or keep track of indeces in an array
* If (++backIndex == QS/ZE)
* backIndex = 0;
* Tree Intro
* Motivation O(N) for lists, average is O(log n)
* Definitions:
* Tree = a finite set of one or more nodes such that:
* There is a specially designated node called the root, r.
* The remaining nodes are partitioned into n >=0 disjoint setes T1, …, Tn, where each of these sets is a tree. We call T1, …, Tn the subtrees of the root. The roots of each of these subtrees are connected by a directed edge from r.
* The root of each subtree is said to be a child of r, and r is the parent of each subtree root. Each node has exactly one parent, the root has no parent.
* More family terminology:
* Siblings (nodes with the same parent)
* Ancestor, descendent
* Leaves = nodes with no children
* A path from n1 to nk (down) is a sequence of nodes n1, n2, .., nk such that ni is the parent of ni+1 for 1 <= I < k.
* Length – number of edges in path, namely k-1
* Everything is a leaf or internal node
* Height is the length of the longest path
* BTree Intro
* Minimize number of disk accesses because those are slow
* Takes about 5 milliseconds (to move harddrive)
* Minimize the height of the tree
* Everything but the root has to be half full