Sorting
1
selection sort:
4 3 1 5 2 -> 1 | 3 4 5 2 -> 1 2 | 4 5 3 -> 1 2 3 | 5 4 -> 1 2 3 4 | 5
o(n^2)

insertion sort:
4 3 1 5 2 -> 3 4 | 1 5 2 -> 1 3 4 | 5 2 -> 1 3 4 5 2 -> 1 2 3 4 5
best: n; average: n^2 worst: n^2

bubble sort:
4 3 1 5 2 -> 3 4 1 5 2 -> 3 1 4 5 2 -> 3 1 4 2 5 | -> 1 3 4 2 5 -> 1 3 2 4 5 -> | 1 2 3 4 5
best:n; average: n^2, worst n^2

2
bubble sort

3
mergeSort(3,7,6,5,8,2,1,4)
merge(mergeSort(3,7,6,5), mergeSort(8,2,1,4))
merge(merge(mergeSort(3,7), mergeSort(6,5)),
merge(mergeSort(8,2), mergeSort(1,4)))
merge(merge(merge(mergeSort(3), mergeSort(7)),
merge(mergeSort(6), mergeSort(5))),
merge(merge(mergeSort(8), mergeSort(2)),
merge(mergeSort(1), mergeSort(4))))
merge(merge((3,7), (5,6)),
merge((2,8), (1,4)))
merge((3,5,6,7), (1,2,4,8))
(1,2,3,4,5,6,7,8)

Hard to write it in-place
CS144 multiple pass merge sort

quickSort(3,7,6,5,8,2,1,4)
(quickSort(2,1), 3, quickSort(7,6,5,8,4))
(quickSort(1),2,3, quickSort(6,5,4), 7, quickSort(8))
(1,2,3,quickSort(5,4), 6, 7, 8)
(1,2,3,quickSort(4),5,6,7,8)
(1,2,3,4,5,6,7,8)

4
find-kth(A, k)
pivot = random element of A
(L, R) = split(A, pivot)
if k = |L|+1, return A[k]
if k ≤ |L| , find-kth(L, k)
if k > |L|+1, find-kth(R, k-(|L|+1))

5
set k = n/2
A more efficient method:
http://cs.stackexchange.com/questions/1914/to-find-the-median-of-an-unsorted-array

Tree

```
Preorder: 5 3 0 2 4 7 6 8 10
Inorder: 0 2 3 4 5 6 7 8 10
Postorder: 2 0 4 3 6 10 8 7 5
Levelorder: 5 3 7 0 4 6 8 2 10 BFS
Someorder: Same as preorder: DFS
```

```
int BinaryTree::numOfNonLeafNodes(Node *node)
{
    if(node == nullptr || (node->left == nullptr && node -> right ==
nullptr))
        return 0;
    return 1+numOfNonLeafNodes(node->left) + numOfNonLeafNodes(node-
>right);
}
```

```
int BinaryTree::height(Node *node)
{
    if (node == nullptr) return 0;
    int left = height(node->left);
    int right = height(node->right);
    return 1+ (left > right ? left : right);
}
```

http://articles.leetcode.com/construct-binary-tree-from-inorder-and-preorder-postorder-traversal/

```
From preorder: Root: 7
From inorder:
Left tree: {4, 10, 3, 1}
     From preorder: Root: 10
     From inorder:
     Left tree: {4}
     Right tree: {3,1}
          From preorder: Root 3
          From inorder: Right tree 1
Right tree: {11, 8, 2} …
```