QUIZ

1. Suppose you are given an Array A[1….n] of sorted integers that has been circularly shifted k positions to the right.
   a. Complexity of insertion sort?
   b. Complexity of quick sort?
   c. Complexity of merge sort?

2. Considering functions f(n) >= 0, g(n) >= 0 and constant b > 0.
   a. Max{f(n), g(n)} = **Θ** (f(n) + g(n)) →FALSE
   b. f(n) + b = O(f(n)) →FALSE

MIDTERM

1. Find 2 functions f(n) and g(n) that satisfy each of relationship. If no such f and g exist, explain why?
   a. f(n) $\in O(g(n))$ and $f(n) \notin \Theta(g(n))$
   b. f(n) $\in \Theta(g(n))$ and $f(n) \in O(g(n))$
   c. $f(n) \in \Theta(g(n))$ and $f(n) \notin O(g(n))$
   d. $f(n) \in \Omega(g(n))$ and $f(n) \notin O(g(n))$

2. Solve recurrence asymptotically
   T(2) = 1
   $T(n) = T(n^{1/2}) + 1$, n > 1

3. Explain briefly how Quick Sort can be modified to run in O(nlogn) time in the worst case.
4. Describe an O(n) algorithm that, given a set S of n distinct numbers and a positive integer k<=n, determines the k numbers in S that are closest to the median of S.
5. Suppose that you are given a sorted array A of n distinct integers, drawn from 1 to m where n <m. That is, A is a subset of [1,2,….,m]. Give an O(logn) time algorithm to find the smallest non-negative integer that is missing in A. Example A[1,2,3,5,7,8,10] return 4.

FINAL

3. **Making Change**

Imagine you live in a country where the coin denominations are 1 cent, 4 cents, and 5 cents. Consider the problem where you are given some value n and you want to make change for this value, using the smallest number of coins.

Part 1: Show that the greedy algorithm (use the largest value coins first) for making change fails for these denominations

Part 2: Describe a dynamic programming algorithm for finding the smallest number of coins needed in this country to make change for any value of n. Analyze your algorithm.

Part 3: Provide the running time of your algorithm.

4. **Amortized Analysis**

Assume you are creating an array data structure that has a fixed size of n. You want to backup this array after every so many insertion operations. Unfortunately, the backup operation is quite expensive, it takes n time to do the backup. Insertions without a backup just take 1 time unit.

Part 1: How frequently can you do a backup and still guarantee that the amortized cost of insertion is $O(1)$?

Part 2: Prove that you can do backups in $O(1)$ amortized time. Use the *potential method* for your proof.

1. Given an unsorted array, the array has this property that every element in array is at most k distance from its position In sorted array where k is a positive integer smaller than size of array. For example let us consider k is 2 an element at index 7 in the sorted array can be at indexes 5,6,7,8,9 in the given array. We can sort such arras more efficiently with the help of Heap data structure. Following is the algorithm that uses a Heap.
   a. Create a Min Heap of size k-1 with first k+1 element

b. One by one remove element from heap, put it in result array, and add a new element to heap from remaining elements.
c. Provide a tight asymptotic upper bound for this algorithm. Show your work.

2. Given an array of jobs where every job has a deadline and associated profit if the job is finished before the deadline. It is also given that every job takes the single unit of time, so the minimum possible deadline for any job is 1. The job sequencing problem is time. Below is an example output for the optimal solution to this problem.

```
Input:  Five Jobs with following
deadlines and profits
JobID   Deadline  Profit
  a       2        100
  b       1        19
  c       2        27
  d       1        25
  e       3        15
Output: Following is maximum
profit sequence of jobs
        c, a, e
```

a. Describe a greedy algorithm for the job sequencing problem.
b. Prove that your greedy choice is optimal (proof by contradiction.)

3. Given numbers in the form of triangle. Starting at the top of the triangle and moving to adjacent numbers on the row below, the problem is to find the max. total from top to bottom. Note that triangles are stores as 2-dimensional arrays.

Maximum path sum in a triangle.

Last Updated: 23-05-2018

We have given numbers in form of triangle, by starting at the top of the triangle and moving to adjacent numbers on the row below, find the maximum total from top to bottom.

Examples :

```
Input :
   3
  7 4
 2 4 6
8 5 9 3
Output : 23
Explanation : 3 + 7 + 4 + 9 = 23

Input :
   8
 -4 4
 2 2 6
1 1 1 1
Output : 19
Explanation : 8 + 4 + 6 + 1 = 19
```