Basic Algorithms (CSCI-UA.0310) Practice Midterm **Professors Abboud, Garg, Oshman, Regev**
Computer Science Department, New York University Fall 2025

NetID:	**Name:**

Instructions (Read every item carefully!)

- **You MUST be seated at your assigned seat. Please record your seat here: _____**
- **Do not open this booklet until you are directed to do so. Read all the instructions on this page.**
- **When the exam begins, write your name on every page of this quiz booklet *before* completing any questions.**
- **Answer each question in the provided box. If you need more space, write in the extra box towards the end of the booklet. We will also provide more scrap paper upon request.**
- **The exam is 75 minutes long. The maximum possible score is 100 points.**
- **Even if you don't know how to solve a problem, show your work and thinking process. We cannot give partial credit for work we do not receive.**
- **You may use any algorithm or theorem we saw in class without proof, as long as you state it correctly. For all questions asking you to give algorithms, you do *not* have to give detailed pseudo-code, or even any pseudo-code. It is enough to give a clear description of your algorithm.**
- **This exam is closed book. No calculators, phones, smartwatches, etc. are permitted. If you need to use the bathroom during the exam, you must leave all such items on your desk.**
- **Do not spend too much time on any one problem. Read them all through first, and attack them in the order that allows you to make the most progress.**
- **Good luck!!**

Name:________________________

1. **(30 Points)** Answer True or False for the following questions. If True, provide a short (one or two sentences) justification. If False, provide a counter-example (if possible) or a short justification.

   Hint: If $f = O(g)$, then $g = \Omega(f)$ and vice versa.

   (a) $\sqrt{n} = O(\log_2 n)$.

   Solution: TRUE / FALSE

   (b) If $f(n) = \Omega(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$.

   Solution: TRUE / FALSE

   (c) If $f(n) = \Omega(g(n))$ and $g(n) = O(h(n))$, then $f(n) = \Omega(h(n))$.

   Solution: TRUE / FALSE

(d) If $\log_2 f(n) = O(\log_2 g(n))$ then $f(n) = O(g(n))$.

Solution: TRUE / FALSE

(e) If $T(1) = T(2) = 1$ and the following recurrence holds for $n > 2$:

$$T(n) = T\left(\frac{n}{2}\right) + n \ ,$$

then $T(n) = \Theta(\log_2 n)$.

Solution: TRUE / FALSE

2. **Divide and Conquer (30 points)** Suppose you are given an array $A[1, \dots, n]$. Note that this array need **not** be sorted. We say that the element $A[k]$ is a **valley** of A if $A[k] \le \min(A[k-1], A[k+1])$. For simplicity, imagine that $A[0] = A[n+1] = \infty$ so that the leftmost and rightmost elements $A[1]$ and $A[n]$ are only compared to $A[2]$ and $A[n-1]$, respectively. Hence, $A[1]$ is a valley if $A[1] \le A[2]$ and $A[n]$ is a valley if $A[n] \le A[n-1]$. Also, assume that n is an integer power of 2.

   (a) Present an $O(n)$ algorithm that finds a valley. Note that there could be multiple valleys in A. Your algorithm needs only to find and return any one of the valleys.

   Solution:

   (b) If $n = 1$, what does your algorithm return? What does the algorithm return if $n = 2$? These are intended to be the base cases for your recursive algorithm based on divide and conquer.

   Solution:

   (c) Suppose you look at $x = A[n/2]$ and $y = A[n/2+1]$. If $x \le y$, then a valley is guaranteed to occur in one of the following halves:

   (1) $A[1, \dots, n/2]$
   (2) $A[n/2+1, \dots, n]$

   Specify which half is guaranteed to contain a valley. Justify your choice.

   Solution: Multiple choice. Check ✓ one of the following two boxes.

   ☐ $A[1, \dots, n/2]$

   ☐ $A[n/2+1, \dots, n]$

   Justify your answer:

(d) Design a recursive algorithm based on Divide-and-Conquer technique to find a valley.

Solution:

(e) Let $T(n)$ be the running time for your Divide-and-Conquer algorithm (from the previous part) on an input array of size n. Derive a recurrence relation for $T(n)$ and write a Θ-expression for $T(n)$.

Solution:

Name:________________________

3. **Dynamic Programming (40 points)** The input is an array $A[1,\ldots,n]$ of length n with integer entries. Our goal is to find the maximum possible sum of the elements of an **even** length sub-array of A. (Recall that a subarray of A is given by $A[i,\ldots,j]$ for some $1 \leq i \leq j \leq n$; in particular, it **cannot** be empty.)

   For example, in the array $[2,-2,10,-3,-20,20]$, the maximum even-length subarray sum is 8, as given by the length 2 subarray $[-2,10]$. The length 4 subarray $[10,-3,-20,20]$ sums to only 7, so it is not the maximum possible. On the other hand, the length 1 subarray $[20]$ sums to 20, but its length is not even.

   (a) How many distinct even-length subarrays are there in an array of length n? It is sufficient to give a Θ expression for this. Give a very brief explanation.

   Solution:

   (b) Let $\mathsf{MaxEven}[i]$ denote the maximum even-length subarray sum ending *at* $A[i]$ (i.e., the right-most element in the subarray is $A[i]$). $\mathsf{MaxEven}[i]$ is defined for $i \in \{2,3,4,\ldots,n\}$. State appropriate base cases for $\mathsf{MaxEven}[i]$.

   Solution:

(c) Derive a recursive formula for $\mathsf{MaxEven}[i]$ in terms of its subproblems.

Solution:

(d) To check your recursive formula, fill out the following table for MaxEven given the following input array:

$$A = [50, 14, -14, 14, -20, -35, 16, 17]\ .$$

Hint: You should still be able to manually fill this table up, even if you did not write the recursive formula.

Solution:

i	2	3	4	5	6
$\mathsf{MaxEven}[i]$					

Name:________________________

(e) Use the previous parts to give a Θ-expression for the running time of an algorithm to fill the array MaxEven. Justify your running time.

Solution:

(f) Assuming that the array MaxEven has been correctly populated, describe what value is returned to solve the problem of finding the maximum even-length subarray sum of $A[1, \ldots, n]$. Also, give a Θ-expression for the *total* running time of your algorithm and provide a short justification.

Solution:

Name:________________________

Extra sheet of paper #1. If you use this sheet, you MUST make a note of it on the original question. OTHERWISE, THIS SHEET WILL NOT BE GRADED.

Solution:

Question number:

Name:________________________

Extra sheet of paper #2. If you use this sheet, you MUST make a note of it on the original question. OTHERWISE, THIS SHEET WILL NOT BE GRADED.

Solution:

Question number:

Name:________________________

Extra sheet of paper #3. If you use this sheet, you MUST make a note of it on the original question. OTHERWISE, THIS SHEET WILL NOT BE GRADED.

Solution: Question number:

Name:________________________

Scratch paper: anything written here will NOT be graded!

Name:________________________

Scratch paper: anything written here will NOT be graded!