C Sc 70010 – Algorithms

Fall 2026 Midterm Exam

October 21, 2026

Student ID (no name): .....

Problem	Maximum Points	Your Points
1(a)	40	
1(b)	10	
2(a)	30	
2(b)	20	
2(b) bonus	10	
Total	100 (+10)	

Some rules:

- You have 2 hours to complete the exam.
- You will get only partial credit if you fail to justify your answers. You will get 20% of the credit if you do not answer a problem. You will get zero credit for wrong answers.

Honor code: Students are expected to take this exam **by themselves** without any external help from other people, the Internet, books, or notes.

Problem 1: Let $A = [A_1 < A_2 < \cdots < A_n]$ and $B = [B_1 < B_2 < \cdots < B_n]$ be two sorted arrays, each containing $n \geq 1$ distinct positive integers, and let k be a positive integer.

Prove that your algorithms are correct and analyze their running times.

Problem 1(a): Design the most efficient algorithm you can that either returns a pair of indices (i, j), with $1 \leq i, j \leq n$, such that $A_i + B_j = k$, or reports that no such pair exists. If several such pairs exist, returning any one of them is sufficient.

An algorithm that runs in $O(n^2)$ time earns up to 15 points, $O(n \log n)$ time up to 25 points, and $O(n)$ time (linear-time) up to 40 points.

Problem 1(b): Design an algorithm that either returns a pair of indices (i, j), with $1 \leq i, j \leq n$, such that $A_i - B_j = k$, or reports that no such pair exists. If several such pairs exist, returning any one of them is sufficient.

Only an $O(n)$-time (linear-time) algorithm earns credit for this part.

Problem 2: For $n \geq 2$, let M be an $n \times n$ matrix containing n^2 distinct positive integers. For $1 \leq i, j \leq n$, the entry in row i and column j of M is denoted by $M[i, j]$. In M, the integers in each row are sorted in increasing order (from left to right) and the integers in each column are sorted in increasing order (from top to bottom). That is, for any three indices $1 \leq k \leq n$ and $1 \leq i < j \leq n$, the following hold:

$$M[k, i] < M[k, j] \quad \text{and} \quad M[i, k] < M[j, k]$$

Let x be a positive integer. The goal is to either find the position (i, j) such that $M[i, j] = x$, or report that x does not appear in M. The cost of an algorithm is the number of comparisons it makes between x and entries of M; all other operations are free. Each comparison of x with $M[i, j]$ has one of three outcomes:

$$x < M[i, j] \quad \text{or} \quad x > M[i, j] \quad \text{or} \quad x = M[i, j]$$

You may assume that n is a power of 2 and may ignore floors and ceilings. You must justify your answers.

Problem 2(a): Describe the most efficient algorithm you can (fewest comparisons in the worst case) that finds the position (i, j) of x in M, or reports that x is not in M.

An algorithm that makes $O(n^2)$ comparisons earns up to 10 points, $O(n \log n)$ comparisons up to 20 points, and $O(n)$ comparisons up to 30 points.

What is the exact worst-case number of comparisons made by your algorithm, as a function of n?

Problem 2(b): Prove that any algorithm that solves this problem, using only comparisons between x and entries of M, must make at least n comparisons in the worst case.

Problem 2(b) – Bonus (10 points): Prove that any algorithm that solves this problem, using only comparisons between x and entries of M, must make at least $2n-1$ comparisons in the worst case.

A correct proof of this bound also proves the bound of n, so it earns the 20 points of part (b) plus the 10 bonus points.

Write your answer to part (a) on page 5, your answer to part (b) on page 6, and your answer to the bonus part on page 7.

Answer to Problem 2(a):

Answer to Problem 2(b):

Answer to the bonus part of Problem 2(b):