Please read these notes

- This exam is **closed book, closed notes** and 50 minutes long
- Read the questions carefully
- No electronic equipment allowed (smart phones, tablets, computers, . . .)
- Write legibly and try to be brief and to the point; what can’t be read will not be graded
- No code: use pseudocode or English to describe your algorithms
- When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class or covered in a CS 218 homework or exam, without giving its details, unless the question specifically requires that you give such details
- Always remember to analyze the time complexity of your solution
- If you have a question, please raise your hand
Problem 1. [Analysis] (25 points)

A CS 218 student has been trying to speed-up Karatsuba’s divide-and-conquer integer multiplication algorithm. Given two numbers $x, y$ with $n$ bits each, his algorithm (1) first divides both $x$ and $y$ into four equal-length pieces, then (2) expresses the product $x \cdot y$ using $p$ multiplications of these $n/4$-bit pieces, followed by a constant number of additions, subtractions and shifts. How small $p$ needs to be in order to give a faster algorithm than the Karatsuba’s algorithm covered in class? You can assume $n$ to be a power of 4, and $p > 4$. Justify your answer.
Problem 2. [Amortized Analysis] (25 points)

Consider an implementation of a dynamic array (similar to vector in C++/STL) which supports operations init and push_back

- A.init() creates an empty dynamic array A with capacity one
- A.push_back(x) stores x in the next available slot in A, if space is available; if the maximum capacity is reached, (1) the capacity is doubled by allocating twice as much memory space, (2) the current data in A is copied in the new allocated space, (3) x is appended at the end of A

Show that n push_back operations on an initialized array A will take $O(n)$ overall.
Problem 3. [Divide and Conquer - Design] (25 points)
Suppose you are given $k$ sorted arrays, each with $n$ elements. Describe a $O(kn \log k)$-time divide-and-conquer algorithm that combines the $k$ sorted arrays into a single sorted array of $kn$ elements. Explain carefully why your algorithm takes $O(kn \log k)$ time.
Problem 4. [Divide and Conquer - Analysis] (25 points)

In the algorithm SELECT described in class (linear time selection), the input elements are divided in groups of 5. Write the recurrence relation for the time complexity of SELECT if you decided to divide the input in groups of 7. You do NOT need to solve the recurrence relation.