Instruction  Put your answers on A4 paper. Your solutions to different problems should be put on separate sheets. The sheets for each problem should be stapled together. Make sure that your name and ID-number are on each problem. Violation of the above rules will cost you one point per problem. We do not accept late submissions.

You are welcome (and encouraged) to discuss with others (including the TAs and the instructor) to solve the problems. You, however, have to write down your solutions using your own words. Copying (even a sentence) from any sources is absolutely prohibited. Most importantly, you have to acknowledge the people and the sources you consulted for each problem on the first page of your solution to that problem. If you solve a particular problem all by yourself, please also say so on the first page of your solution to that problem. For the \( i \)-th homework, we will take \( i \) points away from each solution that has no acknowledgement or a claim of working alone.

Unless you are explicitly asked to prove something that we have seen in class, you may directly use anything shown in class to solve the problems in a black-box manner.

Problem 1  (15 points) We have seen in class the clever \( O(n) \)-time algorithm for the selection algorithm. A critical step in the algorithm is to divide the input numbers into 5-element groups. Now you are asked to analyze the time complexity of the analogous algorithm if we divide the input numbers into 3-element groups. Specifically, you have to give the revised algorithm and show that the time complexity of the algorithm is \( \Theta(f(n)) \) for some function \( f(n) \).

Problem 2  (15 points) We have seen in class the clever \( O(n) \)-time algorithm for the selection algorithm. A critical step in the algorithm is to divide the input numbers into 5-element groups. Now you are asked to analyze the time complexity of the analogous algorithm if we divide the input numbers into 7-element groups. Specifically, you have to give the revised algorithm and show that the time complexity of the algorithm is \( \Theta(f(n)) \) for some function \( f(n) \).

Problem 3  (15 points) We have seen in class the visualization tree for the bubble-sort algorithm with respect to \( n = 3 \) for the purpose for proving the \( \Omega(n \log n) \)-time lower bound for the comparison-based sorting algorithm. Now you are asked to give the visualization tree for the heap-sort algorithm with respect to \( n = 3 \).

Problem 4  (15 points) We have seen in class the visualization tree for the bubble-sort algorithm with respect to \( n = 3 \) for the purpose for proving the \( \Omega(n \log n) \)-time lower bound for
the comparison-based sorting algorithm. Now you are asked to give the visualization tree for the merge-sort algorithm with respect to \( n = 3 \).

**Problem 5**  (40 points) Give asymptotic upper and lower bounds for \( T(n) \) in each of the following recurrences. Assume that \( T(n) \) is constant for sufficiently small \( n \). Make sure that your bounds are as tight as possible, and justify your answers.

- \( T(n) = 3T(n/2) + n \log_{10} n \).
- \( T(n) = 7T(n/7) + n/\log_2 n \).
- \( T(n) = 4T(n/2) + n^{2.7} \).
- \( T(n) = 3T(n/3 + 10) + n/10 \).

**Problem 6**  (30 points) Give asymptotic upper and lower bounds for \( T(n) \) in each of the following recurrences. Assume that \( T(n) \) is constant for sufficiently small \( n \). Make sure that your bounds are as tight as possible, and justify your answers.

- \( T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n \).
- \( T(n) = T(n - 2) + 2 \log_2 n \).
- \( T(n) = T(n - 1) + \log_2 n \).