Problem 1. (20 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. Would the new algorithm still work in linear time? Repeat the analysis if groups of 3 is used.

Problem 2. (40 points)

For $n$ distinct integer $x_1, x_2, \ldots x_n$ with positive weights $w_1, w_2, \ldots, w_n$ such that $\sum_{i=1}^{n} w_i = 1$, the weighted median is the element $x_k$ satisfying

$$\sum_{i : x_i < x_k} w_i < 1/2 \quad \text{and} \quad \sum_{i : x_i > x_k} w_i \leq 1/2$$

Show how to compute the weighted median in $\Theta(n)$ worst-case using a linear-time selection algorithm.
Problem 3. (40 points)

In this problem we are given a text of size $2n$ and a pattern of size $n$ and we want to compute for each position in the text the number of matched symbol with the pattern. A brute force approach would consist in aligning the pattern at each position in the text and computing the number of matches, for an overall $O(n^2)$ time. Here we show that one can solve this problem using the FFT.

Formally, let $t = a_0a_1a_2\cdots a_{2n-1}$ and $p = b_0b_1b_2\cdots b_{n-1}$ be two strings over an alphabet $\Sigma$ of size $\sigma$ (here we assume that $\sigma$ is not a constant). We define an array $C$ as follows $C(i) = \sum_{k=0}^{n-1} \text{equal}(a_{i+k}, b_k)$, for $i \in \{0, 1, \ldots, n\}$, where $\text{equal}(\alpha, \beta)$ is 1 if $\alpha = \beta$, zero otherwise. For example, if $t = \text{abracada}$ and $p = \text{abaa}$ then $C(0) = 3$, $C(1) = 1$, $C(2) = 1$, $C(3) = 2$, $C(4) = 1$. Note that $C(i)$ gives the number of matched symbols when $p$ is aligned with the text at position $i$.

For each $\gamma \in \Sigma$ let us define $C_\gamma(i) = \sum_{k=0}^{n-1} \text{equal}_\gamma(a_{i+k}, b_k)$ for $i \in \{0, 1, \ldots, n\}$, where $\text{equal}_\gamma(\alpha, \beta)$ is 1 if $\alpha = \beta = \gamma$, zero otherwise. Give a $O(n \log n)$ algorithm to compute one of the arrays $C_\gamma$ based on convolution. This implies an efficient algorithm to compute $C$. What is the complexity as a function of $n$ and $\sigma$?