Due 23:59 Feb 11 (Sunday). There are 100 points in this assignment. Submit your answers (**must be typed**) in pdf file to CourSys
`https://coursys.sfu.ca/2024sp-cmpt-307-d1/`.
Submissions received after 23:59 will get penalty of reducing points: 20 and 50 points deductions for submissions received at $[00:00, 00:10]$ and $(00:10, 00:30]$ of Feb 12, respectively; no points will be given to submissions after $00:30$ of Feb 12.

1. 20 points (P 5-2 of text book)

   Searching an unsorted array problem: Given an array of n elements which are not sorted and a value x, find the index i such that $A[i] = x$. A randomized algorithm Random-Search for the problem is as follows: select a number i from $\{1, .., n\}$ independently and uniformly; if $A[i] = x$ then return i and terminate; otherwise repeat the above process until the i with $A[i] = x$ is found or $A[i] \neq x$ for every $i = 1, .., n$ is concluded.

   (a) Write a pseudo-code for Random-Search.

   (b) Assume there is exactly one i such that $A[i] = x$. What is the expected number of checks of $A[i] = x$ Random-Search performs before terminates.

2. 15 points (Ex 6.5-3 of text book)

   Write pseudo-code for procedures Min-Heapify, Heap-Minimum, Min-Heap-Extract, Heap-Decrease-Key, and Min-Heap-Insert that implement a min-priority queue with a min-heap.

3. 15 points (Ex 7.4-5 of text book)

   One variant of quicksort algorithm is that for a small number $k > 1$, the algorithm does not sort any subarray of size smaller than k and returns with the subarray unsorted, and after the top-level call to quick-sort returns, run insertion sort on the entire array to complete the sorting. Prove that this quicksort variant runs in $O(nk + n\log(n/k))$. In theory, how should k be selected?

4. 15 points

   Implement the algorithm Randomized-Quicksort discussed in class and the variant of Randomized-Quicksort given in Question 3 by a same programming language (any language is OK) on a same computing platform. Report the running times of the two algorithms for sorting n numbers with $n = 2^i \times 1000$, $i = 0, 1, 2, 3, 4, 5$. Give your suggestion on selecting k in practice (e.g., the k achieves the best running time in your implementation). Code submission is not needed.

5. 15 points (P 8-2 of text book)

   Given an array A of n elements, each element is 0 or 1, an algorithm for sorting A into increasing order might possess some subset of the following three desirable properties:

   (1) The algorithm runs in $O(n)$ time.

   (2) The algorithm is stable.

(3) The algorithm sorts in place, using no more than a constant amount of storage space in addition to A.

(a) Give an algorithm that satisfies (1) and (2) above.

(b) Give an algorithm that satisfies (1) and (3) above.

(c) Give an algorithm that satisfies (2) and (3) above.

6. 20 points (P 9-3 of text book)

   For n elements $x_1, ..., x_n$ with positive weights $w_1, .., w_n$ such that $\sum_{i=1}^{n} w_i = 1$, we say $x_i < x_k$ if $w_i < w_k$ or $w_i = w_k$ and $i < k$. For $x_1, .., x_n$, the (lower) median is x_k with $k = \lceil n/2 \rceil$ and the weighted (lower) median is the element x_k satisfying $\sum_{x_i < x_k} w_i < \frac{1}{2}$ and $\sum_{x_i > x_k} w_i \leq \frac{1}{2}$. Example: for $x_1, .., x_7$ with $w_1 = 0.1, w_2 = 0.35, w_3 = 0.05, w_4 = 0.1, w_5 = 0.15, w_6 = 0.05, w_7 = 0.2$, the median of $x_1, .., x_7$ is x_4 but the weighted median is x_7.

   (a) Prove that the median of $x_1, .., x_n$ is the weighted median of $x_1, .., x_n$ with weight $w_i = 1/n$ for $1 \leq i \leq n$.

   (b) Show how to compute the weighted median of n elements in $O(n \log n)$ worst-case time using sorting.

   (c) Show how to compute the weighted median in $\Theta(n)$ worst-case time using a linear-time median algorithm such as SELECT from Section 9.3.