Please follow the instructions provided below to submit your assignment.

Worth: 10% **Due:** By 11:59 pm on Friday Sep 29

- You must submit your assignment as a single PDF file through the MarkUs system.

  https://markus.teach.cs.toronto.edu/csc263-2017-09/

  The filename must be the assignment's name followed by "sol", e.g. your submission for the assignment "A1" must be "A1sol.pdf". Any group of two students would submit a single solution through Markus. Your PDF file must contain both team member's full names.
- Make sure you read and understand "POLICY REGARDING PLAGIARISM AND ACADEMIC OFFENSE" provided in the course information sheet.
- The PDF file that you submit must be clearly legible. To this end, we encourage you to learn and use the LaTex typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You can find a latex template in Piazza under resources. You can use other typesetting systems if you prefer. Handwritten documents are acceptable but not recommended. The submitted documents with low quality that are not clearly legible, will not be marked.
- You may not include extra descriptions in your provided solution. The maximum space limit for each question is 2 pages. (using a reasonable font size and page margin)
- For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbook, by referring to it.
- Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision, and conciseness of your presentation.
- For describing algorithms, you may not use a specific programming language. Describe your algorithms clearly and precisely in plain English or write pseudo-code.

1. (20 MARKS) In the following procedure, the input is an array A of size $n \geq 2$ which contains arbitrary integers.

```
procedure ODD-OR-EVEN($A, n$)
    for $i = 2$ to $n$ do
        $x = (i \mod 2)$                    ▹ ($i$ mod 2) is one if $i$ is odd; it is zero, otherwise.
        if $(A[i] + A[i-1]) \mod 2 \neq x$ then
            for $j = i$ to $n$ do
                $A[j]++$
            end for
        else
            return                          ▹ Returns from the procedure call.
        end if
    end for
end procedure
```

Let $T(n)$ be the worst case time complexity of executing procedure ODD-OR-EVEN on an array of size $n \geq 2$. Assume that assignments, comparisons and arithmetic operations, like additions take a constant time each.

(a) (5 MARKS) State whether $T(n)$ is $O(n^2)$ and justify your answer.

(b) (15 MARKS) State whether $T(n)$ is $\Omega(n^2)$ and justify your answer. Any answer without a sound and clear justification will receive no credit.

2. (35 MARKS) Considering the following algorithm which searches for the last appearance of value k in an array A of length n. The index of the array starts from 0.

```
FINDLAST($A, k$):
1   for $i \leftarrow n-1, n-2, \ldots, 0$:
2       if $A[i] = k$:
3           return $i$
4   return $-1$
```

The input array A is generated in the following specific way: for $A[0]$ we pick an integer from $\{0, 1\}$ uniformly at random; for $A[1]$ we pick an integer from $\{0, 1, 2\}$ uniformly at random; for $A[2]$ we pick an integer from $\{0, 1, 2, 3\}$ uniformly at random, etc. That is, for $A[i]$ we pick an integer from $\{0, \ldots, i+1\}$ uniformly at random. All choices are independent from each other. Now, let's analyse the complexity of FINDLAST by answering the following questions. All your answers should be in **exact form**, i.e., **not** in asymptotic notations.

NOTE: For simplicity, we assume that k is an integer whose values satisfies $1 \leqslant k \leqslant n$.

(a) (5 MARKS) In the **best case**, for input (A, k), how many times is Line #2 ("**if** $A[i] = k$") executed? Justify your answer.

(b) (5 MARKS) What is the probability that the best case occurs? Justify carefully: show your work and explain your calculation.

(c) (5 MARKS) In the **worst case**, for input (A, k), how many times is Line #2 executed? Justify your answer.

(d) (10 MARKS) What is the probability that the worst case occurs? Justify carefully: show your work and explain your calculation.

(e) (10 MARKS) In the **average case**, for input (A, k), how many times is Line #2 expected to be executed? Justify your answer carefully: show your work and explain your calculation.

HINT: Your answers should be in terms of both n and k. Thinking about the number of comparisons needed to find a given k, which values are possible, which values are not?

3. (20 MARKS) Given an integer array of n elements, write an algorithm that finds the k smallest elements of the array where k and n are two integers such that $k < n$. You need to give **the most efficient** algorithm for this question. Analyze the time complexity of your algorithm.

4. (25 MARKS) Given an input stream of n integers, we want an efficient algorithm that finds the median of elements that have been read so far. The algorithm must perform the following task for each i^{th} integer:

   - Add the integer to a running list of i^{th} integers and find the median of the updated list.
   - Print the list's updated median after reading each element.

   For example, let us consider the stream $5, 15, 1, 3$

   - After reading the first element of the stream $5 \rightarrow$ median: 5
   - After reading the 2nd element of the stream $5, 15 \rightarrow$ median: 10
   - After reading the 3rd element of the stream $5, 15, 1 \rightarrow$ median: 5
   - After reading the 4th element of the stream $5, 15, 1, 3 \rightarrow$ median: 4

   So, the algorithm output will be $5, 10, 5, 4$.

   The worst case running time of your algorithm must be less than $\Theta(n^2)$. Algorithms with time complexity of $\Theta(n^2)$ and higher will receive no credit.
   HINT: : You need to use the heap data structure for an efficient algorithm.

   (a) (20 MARKS) Describe your algorithm clearly and precisely in plain English.

   (b) (5 MARKS) Analyze the worst case running time of your algorithm.

   REMARK: The *median* is the value separating the higher half of a data set from the lower half. For a sorted data set,

   - if a dataset contains an odd number of elements, the median is the middle element of the sorted sample.
   - if the dataset contains an even number of elements, the median is the average of the two middle elements of the sorted sample.