Please follow the instructions provided below to submit your assignment.

Worth: 10% **Due:** By 11:59 pm on Friday Dec 8

- 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 "A4" must be "A4sol.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. (25 Mark) This question shows the importance of data structures in the study of philosophy. A **paradox** is a group of statements that lead to a contradiction. For example, consider the following group of two statements which form a paradox:

   1) Statement 2 is FALSE.
   2) Statement 1 is TRUE.

   If we assume Statement 1 is TRUE, which says Statement 2 is FALSE, which in turn means Statement 1 is FALSE, therefore we have got a contradiction. If we assume Statement 1 is FALSE, which means Statement 2 is TRUE, which in turn means Statement 1 is TRUE, again we have got a contraction. That is, assuming one statement to be TRUE or FALSE will lead to deriving a conclusion that is opposite to the assumption.

   Now you are given a group of N statements, numbered from 1 to N. Each statement has the format "`Statement X is TRUE/FALSE`" where `X` is a number between 1 and N. Your task is to figure out whether this group of statements form a paradox. In particular, answer the following questions.

   (a) Apparently you are supposed to use a graph. How do you construct the graph for solving this problem?

   (b) What property must the graph have if the N statements do form a paradox? What property must the graph have if the N statements do NOT form a paradox? Justify your answer.

   (c) How do you efficiently detect whether the graph has the properties you described? Describe your algorithm in concise English.

   (d) What is the worst-case runtime of your algorithm?

2. (25 Marks) Assume that you are a kindergarten teacher, who is given a task to arrange n ill-behaved children in a straight line, facing front. You are given a list of m statements of the form "kid i hates kid j". If kid i hates kid j, then you do NOT want to put i somewhere behind j, because then i is capable of throwing something at j.

   Design an algorithm that efficiently computes a way of arranging these kids so that no throwing can happen, if such an arrangement is possible at all. In particular, answer the following questions.

   (a) How do you model this problem using a data structure that we have studied?

   (b) Is it always possible to find a valid arrangement? If yes, briefly explain why. If no, briefly explain what is the case in which it is not possible.

   (c) How does your algorithm work? And what is its worst-case runtime (in asymptotic notation)? Explain in concise English.

   (d) Suppose instead you want to arrange the children in *rows* such that if kid i hates kid j, then kid i must be in a lower numbered row than kid j (they cannot be in the same row). **Design** an efficient algorithm to find the minimum number of rows needed, if the arrangement is possible. **Explain** in clear English the design of your algorithm and its worst-case runtime (in asymptotic notation). Write pseudo-code if it helps explaining.

   Hint: Start from the straight line you got from the previous arrangement.

3. (25 Marks) Consider the abstract data type DEPR that consists of a set S of positive integers. Initially, S is a set of n consecutive integers. $S = \{a, ..., b\}$ such that $b = a + n - 1$. DEPR supports just the following two operations:

   - DELETE(S, i): Delete integer i from the set S. If $i \notin S$, there is no effect.
   - PREDECESSOR(S, i): Return the predecessor in S of integer i, i.e. $\max\{j \in S \mid j < i\}$. If i has no predecessor in S, i.e. if $i \leq \min S$, then return 0. Note that it is not necessary for i to be in S.

   For example, if $a = 1, n = 7$ and $S = 1, 3, 6, 7$, then PREDECESSOR$(S, 1)$ returns 0, PREDECESSOR$(S, 2)$ and PREDECESSOR$(S, 3)$ return 1.

   Describe a data structure with $O(log^* m)$ amortized cost per operation, where m is the number of operations that are performed. Justify the correctness and time complexity of your data structure.

   How do you initialize your data structure and how much time does it take?

4. (10 Marks) Consider the problem of finding the minimum spanning tree connecting n distinct points in the plane, where the distance between two points is the ordinary Euclidean distance. In this problem we assume the distances between all pairs of points are distinct. For each of the following procedures, either argue that it constructs the minimum spanning tree of the points or give a counterexample.

   (a) Sort the points in order of their x-coordinate. (You may assume without loss of generality that all points have distinct y-coordinates; this can be achieved if necessary by rotating the axes slightly.) Let $(p_1, p_2, ...p_n)$ be the sorted sequence of points. For each point p_i, $(2 \leq i \leq n)$ connect p_i with its closest neighbor among $p_1, ..., p_{i-1}$.

   (b) Draw an arbitrary straight line that separates the set of points into two parts of equal or nearly equal size (i.e. within one). (Assume that this line is chosen so it doesn't intersect any of the points.) Recursively find the minimum spanning tree of each part, then connect them with the minimum-length line segment connecting some point in one part with some point in the other (i.e. connect the two parts in the cheapest possible manner).

5. (15 Marks) Let G be a connected undirected weighted graph. Assume that the weights are distinct meaning that no two edges has the same weight. Prove that the graph has a unique MST.