Six questions, marked out of a total of **70 marks**.

For submitting your assignment, please read and follow the instructions given in course homepage otherwise, your submission will not be considered as a valid submission for the marking process. (http://cgi.cse.unsw.edu.au/ cs3121/assignments.php)

1. (a) **[5 marks]** Describe an $O(n \log n)$ algorithm (in the sense of the worst case performance) that, given an array S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x. (5 pts)

   (b) **[5 marks]** Describe an algorithm that accomplishes the same task, but runs in $O(n)$ expected (average) time.

   Note that brute force does not work here, because it runs in $O(n^2)$ time.

2. **[10 marks]** You're given an array of n integers, and must answer a series of n queries, each of the form: "how many elements of the array have value between L and R?", where L and R are integers. Design an $O(n \log n)$ algorithm that answers all of these queries. (10 pts)

3. **[10 marks]** There are N teams in the local cricket competition and you happen to have N friends that keenly follow it. Each friend supports some subset (possibly all, or none) of the N teams. Not being the sporty type – but wanting to fit in nonetheless – you must decide for yourself some subset of teams (possibly all, or none) to support.

   You don't want to be branded a copycat, so your subset must not be identical to anyone else's. The trouble is, you don't know which friends support which teams, so you can ask your friends some questions of the form "Does friend A support team B?" (you choose A and B before asking each question). Design an algorithm that determines a suitable subset of teams for you to support and asks as few questions as possible in doing so.

4. **[10 marks]** Given n numbers $x_1, \ldots, x_n$ where each x_i is a real number in the interval $[0, 1]$, devise an algorithm that runs in linear time that outputs a permutation of the n numbers, say $y_1, \ldots, y_n$, such that $\sum_{i=2}^{n} |y_i - y_{i-1}| < 2$. *Hint: this is easy to do in $O(n \log n)$ time: just sort the sequence in ascending order. In this case, $\sum_{i=2}^{n} |y_i - y_{i-1}| = \sum_{i=2}^{n} (y_i - y_{i-1}) = y_n - y_1 \leq 1 - 0 = 1$. Here $|y_i - y_{i-1}| = y_i - y_{i-1}$ because all the differences are non-negative, and all the terms in the sum except the first and the last one cancel out. To solve this problem, one might think about tweaking the* BucketSort *algorithm.*

5. You are at a party attended by n people (not including yourself), and you suspect that there might be a celebrity present. A *celebrity* is someone known by everyone, but does not know anyone except herself/himself. (Of course everyone knows herself/himself).

Your task is to work out if there is a celebrity present, and if so, which of the n people present is a celebrity. To do so, you can ask a person X if they know another person Y (where you choose X and Y when asking the question). (10 pts)

(a) **[10 marks]** Show that your task can always be accomplished by asking no more than $3n - 3$ such questions, even in the worst case.

(b) **[5 marks]** Show that your task can always be accomplished by asking no more than $3n - \lfloor \log_2 n \rfloor - 2$ such questions, even in the worst case.

6. You are conducting an election among a class of n students. Each student casts precisely one vote by writing their name, and that of their chosen classmate on a single piece of paper.

   However, the students have forgotten to specify the order of names on each piece of paper – for instance, "Alice Bob" could mean Alice voted for Bob, or Bob voted for Alice!

   (a) **[2 marks]** Show how you can still uniquely determine how many votes each student received.

   (b) **[1 mark]** Hence, explain how you can determine which students did not receive any votes. Can you determine who these students voted for?

   (c) **[2 marks]** Suppose every student received at least one vote. What is the maximum possible number of votes received by any student? Justify your answer.

   (d) **[7 + 3 = 10 marks]** Using parts (b) and (c), or otherwise, design an algorithm that constructs a list of votes of the form "X voted for Y" consistent with the pieces of paper. Specifically, each piece of paper should match up with precisely one of these votes. If multiple such lists exist, produce any. An $O(n^2)$ algorithm earns 7 marks, and an $O(n)$ algorithm earns an additional 3 marks.

   Hint: first, use part (c) to consider how you would solve it in the case where every student received at least one vote. Then, apply part (b).