This assignment is for both COMP3027 and COMP3927 students.

Task 1: Sort of Sorted Arrays [50 marks]

Your genius friend discovered an amazingly fast sorting algorithm using deep quantum neural networks. Unfortunately, due to complicated quantum physics beyond the scope of this unit, the algorithm's output is only sort of sorted. An array S of n integers is *sort of sorted* if for some x, $S[i]$ is the $(x+i \mod n)$-th smallest number. (Note that 0-th smallest number is the smallest and the $(n-1)$-th smallest is the largest.)

For example, $[2, 3, 4, 5, 1]$ is a sort of sorted array with $x = 1$ and $[4, 5, 1, 2, 3]$ is sort of sorted with $x = 3$.

You are given a sort of sorted array S containing n distinct integers, but you are not given x. Your task is to design an $O(\log n)$-time divide-and-conquer algorithm that finds the largest number in S.

(a) Description of how your algorithm works ("in plain English"). Make sure to clearly define your divide step (i.e. subproblems and base cases) and merge step.

(b) Prove that your algorithm is correct.

(c) Prove an upper bound on the time complexity of your algorithm.

(d) Implement your algorithm on Ed.

Task 2: Gotta Catch the Most [50 marks]

After a hard day's work, you relax by playing Pokemon Go. You only have enough time to visit one Pokestop and want to find one with the most Pokemon in its vicinity.

We now state the problem more formally. You are given a set S of n_s non-overlapping squares (representing the area near a Pokestop) with a common side length L and a set P of n_p distinct points (representing Pokemon). Every point has integer coordinates, L is an integer, and the corners of every square have integer coordinates. Let $n = n_s + n_p$. Your task is to design an $O(n \log n)$-time divide-and-conquer algorithm that finds the square that contains the most number of points.

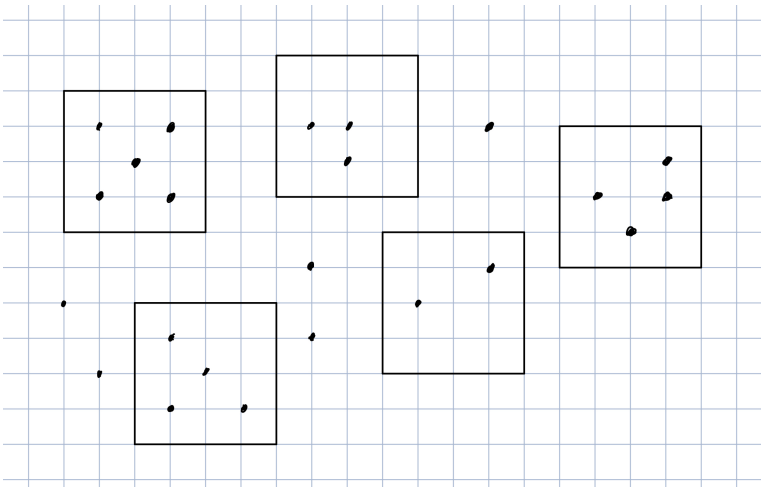

Figure 1: The left-most square has the most points.

(a) Description of how your algorithm works ("in plain English"). Make sure to clearly define your divide step (i.e. subproblems and base cases) and merge step.

(b) Prove that your algorithm is correct.

(c) Prove an upper bound on the time complexity of your algorithm.

Here are some guidelines to make it easier for you to write and for us to mark:

- Assume that you can check in $O(1)$ time whether a square s contains a point p, whether a point p is on a line ℓ, and whether a line ℓ passes through a square s. **Please do not describe your own methods for these simple checks.** For instance, you can simply write "if square s contains point p, ..." without further explanation.
- Assume that you can sort n squares and points in $O(n \log n)$ time. **Please do not describe your own sorting algorithm.** For instance, you can simply write "sort squares in ascending order of their center's y-coordinate"

Submission details

- **Submission deadline is Wednesday 31 March, at 23:59.** Late submissions without special consideration will be subject to the penalties specified in the first lecture (20% per day). **Submissions later than Friday 2 April, 23:59 will not be accepted.**
- Submission time is the max of the submission times to Gradescope and Ed. For example, if you submit to Gradescope on time but your implementation on Ed is one day late, 20% will be deducted from the entire assignment.
- Submit your answers as a single document to Gradescope. Your work must be typed (no images of text, although you can use diagrams if you think it helps.) Please try to be reasonably concise (one to two pages of a4).
- The implementation required for Task 1 should be done in Ed, and submitted via Ed.
- Your report will be subject to automatic and manual plagiarism detection systems. Remember, it's acceptable to discuss high level ideas with your peers, but you should not share the detail of your work, such as parts of the precise algorithms, examples, proofs, writing, or code.
- To facilitate anonymous grading, please do not write your name on your submission.