1. **(3 marks)** Recall the definition of the *iterated logarithm* function: $\log^* n$ is the number of times that you have to take the log of n for it to be at most 1.

   Give the tightest asymptotic relation between the following two functions.

$$f(n) = \log(\log^* n), \text{ and}$$
$$g(n) = \log^*(\log n).$$

> **Solution:** From the definition of $\log^* n$, we can write $g(n) = \log^*(\log n) = (\log^* n) - 1 = \Theta(\log^* n)$. Since $f(n) = \log(\log^* n)$ and the function $\log^* n$ is monotonically non-decreasing, we can conclude that $f(n) = o(g(n))$.

2. **(3 marks)** Consider the following statement: "In every instance of stable matching between courses and students, if the Gale-Shapley algorithm is run with the students applying to courses, at least one of the students will be alloted a course that is his/her first preference".

   If this statement is true, prove it. Otherwise, provide a counter-example.

> **Solution:** Consider the following instance of stable matching
>
> $S_1 : C_1, C_2, C_3$ $C_1 : S_2, S_3, S_1$
>
> $S_2 : C_2, C_1, C_3$ $C_2 : S_3, S_1, S_2$
>
> $S_3 : C_1, C_3, C_2$ $C_3 : S_1, S_2, S_3$
>
> If we start with S_1 applying to C_1, and S_2 applying to C_2, both are accepted. When S_3 applies to C_1, S_1 is rejected. S_1 applies to C_2 and S_2 is rejected. S_2 applies to C_1 and S_3 is rejected, which applies to C_3. So the stable matching is $(S_1, C_2), (S_2, C_1), (S_3, C_3)$.

3. Show that squaring an $n \times n$ matrix is no easier that computing the product of two $n \times n$ matrices in the following way: Suppose that there is a $T(n)$-time algorithm for squaring an $n \times n$ matrix, where $T(n) = O(n^c)$, for $c \geq 2$.

   (a) **(2 marks)** Given two $n \times n$ matrices A and B, show that you can compute $AB + BA$ in time $3T(n) + O(n^2)$.

> **Solution:** We can write $AB + BA = (A + B)^2 - A^2 - B^2$ and since time for squaring a matrix is $T(n)$, we get the running time as $3T(n) + O(n^2)$.

   (b) **(2 marks)** Use the part above to show that given two $n \times n$ matrices X and Y, we can compute XY in time $3T(2n) + O(n^2)$, and hence conclude that multiplying two matrices also takes time $O(n^c)$.

> **Solution:** Consider the following $2n \times 2n$ matrices
>
> $$A = \begin{pmatrix} Y & 0 \\ 0 & 0 \end{pmatrix}, \text{ and } B = \begin{pmatrix} 0 & 0 \\ X & 0 \end{pmatrix}$$
>
> Now, we can verify that
>
> $$AB + BA = \begin{pmatrix} 0 & 0 \\ XY & 0 \end{pmatrix}$$
>
> So, we can apply the algorithm in Part (a) to get the product of X and Y in time $3T(2n) + O(n^2)$

4. **(5 marks)** Suppose you are given an array $A[1, 2, \ldots, n]$ of elements such that you are allowed queries of the form "Is $A[i] < A[j]$" and "Is $A[i] = k$". Give an $O(n)$-time algorithm to check if there is an element that occurs at least $n/4$ times. You cannot assume that the elements in the array are numbers.

   Hint: This is slightly different from the question in the problem set. Try using order statistics.

   Solution: Consider the k^{th} smallest elements where $k = in/5$ for $1 \leq i \leq 4$. We are taking $n/5$ to avoid any funny business that might arise due to floors and ceilings.

   Now, if there is an element in the array that occurs at least $n/4$ times, it must be one of these four elements. We can prove this by contradiction. Suppose not; then any other element will lie between $in/5$ and $(i+1)n/5$ of which there are only at most $n/5 < n/4$ elements.

   Hence we use the median-of-medians algorithm taught in class four times with $k = in/5$, where $1 \leq i \leq 4$ to obtain these elements. After this, we scan the array to count the number of occurrences of these elements, and answer yes if at least one of the occurs more than $n/4$ times. The total running time will be $O(n)$.