P1. (20 points) *(This problem is given to enrich your knowledge on sorting)*

For each of the following sorting methods, give a brief sketch of the algorithm and time complexity.

- Counting Sort
- Bucket Sort
- Radix Sort

What is a “stable” sorting algorithm? Why is stable sorting important? Which of the above three sorting methods are “stable”?

How is Heapsort paradigmatically different from the sorting algorithms described above? *(The answer is a one-liner).*

P2. (15 points)

Solve the recurrence relation *without* using Master’s theorem:

\[ T(N) = 3T(N/2) + cN \]

P3. (10 points)

\[ A \] and \( B \) are playing a guessing game where \( B \) first thinks up an integer \( X \) (positive, negative or zero, and could be of arbitrarily large magnitude) and \( A \) tries to guess it. In response to \( A \)’s guess, \( B \) gives exactly one of the following three replies:

a) Try a bigger number
b) Try a smaller number or
c) You got it!!

Design an efficient algorithm to minimize the number of guesses \( A \) has to make. An example (not necessarily an efficient one) below:

<table>
<thead>
<tr>
<th>A’s guess</th>
<th>B’s response</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Try a bigger number</td>
</tr>
<tr>
<td>20</td>
<td>Try a bigger number</td>
</tr>
<tr>
<td>30</td>
<td>Try a bigger number</td>
</tr>
<tr>
<td>40</td>
<td>Try a smaller number</td>
</tr>
<tr>
<td>35</td>
<td>You got it</td>
</tr>
</tbody>
</table>
P4: (10 points)

There are $n$ students in a class. Assume that the scores received by each student is distinct. You can think of the scores as an unsorted integer array. A student $X$ has been told that his rank in the class is $R$ ($R$ is an integer and obviously, $1 \leq R \leq n$). He wants to find out the $k$ students who are ranked closest to him ($k/2$ students below him, and $k/2$ students above. You may assume that $k$ is even). Devise an efficient algorithm to identify the scores of these $k$ students.

P5. (10 points)

Use equivalence (≡) and less than (<) signs to list the functions in increasing asymptotic order:

a. $(1.5)^n \quad 3^{(n/2)} \quad 2^{(n/3)}$

b. $\lg n \quad \ln n \quad \log n$ (bases 2, $e$ and 10)

c. $\lg(\lg n) \quad \sqrt{(\lg n)} \quad \lg \sqrt{n}$

d. $n^{\lg 4} \quad 2^{\lg n} \quad 2^{2\lg n}$

(Like, we can say that $n < n \log n \equiv 2n + 3n \log n < 2n^2$)

P6. (15 points)

According to the NYSE rules, share prices are announced at the beginning of every minute and the announced price is the value of the share for the next minute. (No intermediate prices are available). Also, in order to bring some sanity in the market after the dotcom meltdown, a rule has been imposed that share prices may rise or fall only by a cent per minute (no fractional change or more than 1 cent change is allowed). However, it may also remain constant for that minute interval.

You have a graph of Amazon.com share value over the day (with minute and cents as $x$ and $y$ axis respectively). You want to find out all the instances of time when the price of the share was exactly $z$ cents (there could be three cases: one instance, multiple instance or none instance of time when the share was priced at $z$ cents).

Design an efficient Divide-and-Conquer algorithm for solving this problem and analyze its complexity for the 3 cases. Is there any algorithm which can boast of $O(\lg n)$ time even in the worst case?
P7. **(15 points)**

Consider the following problem. You have an \( n \times n \) matrix \( A[i, j] \), \( 1 \leq i, j \leq n \). Suppose that each row and column of \( A \) is sorted. I.e., for a fixed \( i \), \( A[i, j] \) increases with \( j \) and for a fixed \( j \), \( A[i, j] \) increases with \( i \).

Input: a number \( x \).
Output: “yes” if \( x \) is in \( A \), “no” otherwise.

Design an efficient Divide-and-Conquer algorithm for this. Explain how your algorithm sticks to the essence of divide-and-conquer algorithm, i.e., how it actually divides the input set and conquers. (You may use diagram to prove your argument). The algorithm alone (without the explanation why it is Divide-and-Conquer) will not get full credit.

P8. **(5 points)**

Space complexity is a measure of how much space the algorithm would use (for reading or writing). Both are expressed similarly in terms of input size \( n \). Sometimes it is better to specify both the time and space complexities of an algorithm to understand its performance (like say an algorithm takes \( O(n^2) \) time and \( O(n) \) space). Which of the following statements is correct, and why? (hint: it takes time to read or write)

- Time complexity = Space complexity
- Time complexity \( \geq \) Space complexity
- Time complexity \( \leq \) Space complexity