This is an individual/group homework assignment. Groups of **up to 2** students can submit joint solutions. **EXACTLY** one student in the group should submit the solution document, and at the beginning of the document, you should clearly **state the names of all groupmates**. The student who submits the solution on behalf of the group is responsible for sharing the grading feedback with the rest of the group. If a groupmate's name is missing on the submitted file, that student will NOT receive any points for the corresponding assignment.
No credit will be given for handwritten solutions (except for figures or long equations that might be handwritten). You must submit your assignment on Canvas in the related digital repository by the specified deadline.
As it appears in the course syllabus, for the homework assignments, students are encouraged to discuss the problems with others, but you are expected to turn in the results of your own effort (not the results of a friend's efforts in another group or so). Even when not explicitly asked, you are supposed to justify your answers concisely and clearly.

Question 1)
Solve the following recurrence relations. For each one come up with a **precise** function of n in closed form (i.e., resolve all sigmas, recursive calls of function T, etc) using the substitution method. Note: An asymptotic answer is not acceptable for this question. Justify your solution and show all your work.

a) T(n)=T(n-1)+cn, T(0)=1,
b) T(n)=4T(n/2)+n , T(1)=1
c) T(n)= 2T(n/2)+1, T(1)=1

Question 2)
Consider Question 1 again. Apply Master Theorem if applicable for each case. Bound the recurrence relation in Big-O.

Question 3)
A binary tree's "maximum depth" is the number of nodes along the longest path from the root node down to the farthest leaf node. Given the root of a binary tree, write a complete program in C++/Java that returns three's maximum depth. What is the time-complexity of your algorithm in the worst-case once you have n nodes in the tree. Analyze and clearly discuss your reasoning. Paste your complete program in the solution file.

Question 4)
Part (a) Write a **linear time divide and conquer algorithm** (i.e., $\theta(n)$) to calculate x^n (x is raised to the power n). Assume a and n are >=0.
Part (b) Analyze the time complexity of your algorithm in the worst-case by first writing its recurrence relation.

Part (c) Can you improve your algorithm to accomplish the end in O(log n) time complexity (we still look for a divide and conquer algorithm). If yes, write the corresponding algorithm, write the recurrence relation for its time complexity and analyze it. If no, justify your answer.