Q1. Write a *dynamic programming* algorithm for computing $M(1,n)$ from the following formula. Analyze the complexity for your algorithm. Drawing a table for $M$ is necessary. $M(1, i)$ are given for all $i$ as input.

$M(i, j) = 0$, for all $i > j$

$M(i, j) = \max \{ M(i, k) + M(k, j) + 2 \mid \text{for all } k \text{ with } i < k < j \}$,

for all $1 \leq i < j \leq n$
Q2. Write a recursive *divide-and-conquer* algorithm for computing a sequence of alternating addition and subtraction of numbers, e.g., a-b+c-d+e-f. Analyze its space & time-complexity (presume input size is some power of 2). [20]
Q3. Depth First Search algorithm for a graph:
Input: graph G=(V, E);  Output: a DFS spanning tree over G

DFS(node v)

(2a) Write the DFS algorithm to print post-order numbering of nodes.  [10]

(2b) Draw an undirected G=(V={a, b, c, d, e}, E={(a,b), (b,c), (b,e), (b, d), (c,e), (c,d) }).
Starting with a call to DFS(a), show your call sequences (i.e., the recursion tree or the
traversal of the graph) and your post-order numbering of the nodes.  [10]
Q4. Suppose a test has 4 questions \{q_1, q_2, q_3, q_4\}, each question number \(q_i\), is associated with \(p_i\) points, and \(t_i\) time-needed-to-answer, which are like the following \{(q_1, p=2, t=2), (q_2, p=3, t=2), (q_3, p=5, t=2), (q_4, p=1, t=2), (q_5, p=5, t=2)\}. Partial grading is allowed, i.e., one gets points proportional to the time spent in answering a question.

Maximum time for the test is \(T=7\). Find best set of questions to answer by using a greedy algorithm.

Both the optimum set of questions and the corresponding optimum aggregate points must be computed.

[20]
**Q5a.** What is the value of the variable count in terms of $n$ after the following algorithm-fragment is executed? \[10\]
(1) count = 0;
(2) For $i = 1$ through 3 do
(3) For $p = 1$ through $i^2$ do
(5) For $k = 1$ through 5 do
(4) count = count +1;
end for loops;

**Q5b.** What is the time complexity of the following algorithm fragment in terms of $n$? \[10\]
(0) int count := 0;
(1) For $i = 1$ through $n$ do
(2) For $p = 1$ through $4*i$ do
(3) For $k = 1$ through $i$ do
(4) count++; end for loops;
(5) print count;