1. Set up the recurrence equation for asymptotic time complexity of the following algorithm and solve it for the usual theta function. [Ignore the purpose of the algorithm.]

Algorithm Little (int array A[], int start, int end)
begin
if end = = start do
    return start;
else
    int x = start +1; // constant time operation
    Little (A, x, end);
    Little (A, start, end-1);
end algorithm.

2a. Explain in a line or two the time complexity of the following algorithm-fragment in terms of \( n \).
\begin{enumerate}
    \item For \( i = 1 \) through \( n \) do
    \item For \( j = 3 \) through \( i \) do
    \item -constant number of steps-
end for loops;
\end{enumerate}

The following is a directed weighted graph. Draw it first. [Usual presumption of adjacency list representation of the graphs holds for all graph theoretic questions.]
\( V=\{a, b, c, d, e\}, E=\{(a, b, 2), (a, d, 8), (b, c, 3), (c, d, 2), (c, e, 5), (d, e, 1), (e, b, 2)\}. \)

2b. After running the following algorithm fragment on this graph show the output for the variable \( count \). Explain your answer in a line or two.
\begin{enumerate}
    \item int \( count := 0; \)
    \item For each node \( v \) in \( V \) do
    \item for each edge \( (u, w, d) \) in \( E \) do
    \item \( count++; \)
end for loops;
\end{enumerate}

3a. For the following algorithm find out what the value for \( count \) is. Explain your answer in a line or two. [Use graph from question 2b.]
\begin{enumerate}
    \item int \( count := 0; \)
    \item For each node \( v \) in \( V \) do
    \item for each edge \( (u, w, d) \) in \( E \) do
    \item if \( v = = u \) then \( count++; \)
end for loops;
\end{enumerate}
3b. For the following algorithm find out what the output from line 4 would be. [Use graph from question 2b.]

(0) int count := 0;
(1) enqueue all arcs in Q;
(2) while Q not empty do
(3) \((v, w, d) = pop(Q)\);
(4) print \((v, w, d)\);
(5) \(d = d - 5\);
(6) if \(d >= 0\) then push \((v, w, d)\) on Q;
end while loop;

4. Write a dynamic programming algorithm for computing \(C(1,n)\) from the following formula. Analyze the complexity for your algorithm.
Input to the algorithm: a matrix of integers \(p_{ij}\), \(1 \leq i \leq n\), \(1 \leq j \leq n\), for problem size \(n\).
\[C(i, j) = 0, \text{ for all } 1 \leq j < i \leq n.\]
\[C(i, j) = \min\{ C(i+k1, j) + p_{ij}, C(i, j-k2) - p_{ij} \mid \text{for all } k1, k2 \text{ with } 1 \leq k1 \leq n-i, 1 \leq k2 \leq j\},\]
\[\text{for all } 1 \leq i \leq j \leq n\]

5. Answer true/false for the following sentences (answer on the question paper):

a. Sets of NP-hard problems and NP-complete problems have null intersection.
b. The set of P-class problems is a subset of the NP-class of problems.
c. NP-complete problems cannot have polynomial algorithms is a conjecture.
d. In order to prove a problem \(X\) to be NP-hard one needs to develop a polynomial transformation from \(X\) to a known NP-hard problem.
e. 4-SAT (where each clause in a Boolean Satisfiability problem has four literals) is an NP-hard problem.