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. For the purpose of analysis assume problem size n is a power of 2.]

Algorithm Little (int array A[], int start, int end)
begin
if end <= start do
    return start;
else
    int x = (start + end)/2; // constant time operation
    Little (A, x, end);
    Little (A, start, x-1);
end algorithm.
2a. What is the value of the variable `count` in terms of `n` after the following algorithm-fragment is executed?
(1) `count = 0;`
(2) For `i = 1` through `n` do
(3) For `p = 1` through `3` do
(4) For `k = 1` through `i` do
(5) `count = count +1;`
end for loops;

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.
(0) `int count := 0;`
(1) For each node `v` in `V` do
(2) for each edge `(u, w, d)` in `E` do
(3) `count++;`
end for loops;
(4) print `count;`
3. Given a set of Boolean variables $V = \{v_1, v_2, \ldots, v_n\}$ and a propositional formula over those variables, in a conjunctive normal form $C = \{c_1, c_2, \ldots, c_m\}$, where $c_i$ is a clause, finding if there exists an assignment for the variables satisfying the formula is called a SAT problem. Write a backtracking algorithm to solve a given SAT problem. What is the asymptotic time-complexity of the algorithm?
4. Write a *dynamic programming* algorithm for computing $C(1,n)$ from the following formula. Analyze the complexity for your algorithm.

$C(i, j) = 0$, for all $i=0$ or $j=0$

$C(i, j) = \min \{ C(i-1, j) + 2, C(i, j-2) - 2 \}$,

for all $1 \leq i \leq j \leq n$
5. Answer very briefly – in a word or a sentence.

a. Name an algorithm for finding shortest path on a weighted graph from a given starting node.

b. Name an algorithm for finding the shortest spanning tree on a graph.

c. What is the minimum asymptotic time complexity of comparison-based sorting algorithms?

d. Name an algorithm which is NOT comparison based and whose complexity is lesser than that of comparison-based sorting algorithm.

e. 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 $Y$ or other way round?

f. Is 2-SAT an NP-class problem?

g. Why is the space complexity of an algorithm always less than or same as that of time-complexity?

h. Self-referencing is not allowed in logic. E.g., “This sentence is true.” - has neither True nor False value. What is the implication of this in Computer Science?

i. What is the number of arcs on a tree with $m$ nodes?

j. Is SAT problem a P-class problem?