Question 1

(a) Describe a polynomial-time reduction from BoxDepth to Clique
In order to reduce the given BoxDepth problem into a Clique problem we need to reduce it to a graph-like problem. We create a graph where each node represents a Rectangle in the BoxDepth problem. Two nodes are connected by an edge if and only if the two rectangles intersect, i.e. they have a common x,y coordinate. More formally, This new graph G will have n vertices indicating the n rectangles and out of these, we have k <= n rectangles which intersect and are connected. The time taken to perform this reduction would be O(n^2) as we need to compare for intersections between each rectangle pair. The k nodes can now form a clique.

Conversely, Suppose we have a Clique in Graph G, where each node is connected to the other in this clique. Now if these nodes are considered to be a rectangle then the k rectangles intersect pairwise in a nonempty rectangle. The x coordinates of this non-empty rectangle will be the minimum of right and max of left on the x-axis. Similarly for the y-axis. To prove by induction if we add a k+1th rectangle to this. Its intersection with any rectangle r <= k would have a similar property. By these min and max properties, we can observe that now we k+1 rectangles intersecting into a non-empty rectangle. Hence the two problems go both ways.

(b) Describe (in English) a polynomial-time algorithm for BoxDepth and analyze its runtime

Suppose we have n rectangles then there can be 2n projection points plotted on the x-axis and 2n points on the y-axis. If we convert these points into intervals we get 2n + 1 intervals on both axes. There can be some coinciding points but for getting the upper bound we assume that we have all distinct points.
Now we check each point one by one from our (2n+1) * (2n+1) regions. (each region can be represented in a 2D-Array) against the points of n rectangles one by one. We are looking if any interval regions are intersecting with each rectangle (finding a common point).
We then iterate and compute the maximum.
The total time taken is O(2n+1)*(2n+1)*(1)*(n) which can be written as O(n^3).

This can be optimized to O(nlogn) by sorting points on one of the axes. and then traversing the points one by one. As we encounter the points we get its counterpart on the other axis and maintain a augmented Binary search tree keeping track of the highest interval. The tree operations are Log(n) and hence upper bound would be the O(nlogn).

(c) Why don't these two results imply that P = NP?
We know that BoxDepth is solvable in polynomial time in the 2nd part. In the first part, we reduced BoxDepth to Clique, i.e. P was reduced to an NP-hard problem. Suppose if we would have converted an NP-complete problem to a P problem, we could say that P = NP. Since we did the reverse, that is converted an easy problem to a hard one, we can't make such conclusions.

Question 2

We can prove that this is an NP-hard problem by reducing SAT problem, which we already know is NP-hard to this problem. For simplicity of proofs we can consider 3SAT for now.
Let us consider ϕ as a CNF boolean formula representing a 3SAT problem consisting of literals and clauses. We need to reduce this formula to the Snowman/Snowflake problem given. We are given that the size of the grid is n*m, i.e. we have n rows and m columns. Let each row be represented as clauses and columns as literals. Now we are given conditions to be satisfied and for all the indices i and j:

(a) if the jth literal (xj) of the ith clause is set then we can place a snowman at i,j.

(b) if the negation of jth literal ($\bar{xj}$) is in the ith clause then we place a snowflake at i,j.

(c) else insert blank at i,j

We can say that the puzzle is solvable if and only if the formula is satisfiable.
Assume that ϕ is satisfiable. Consider an assignment of literals in the formula. For the formula to be satisfiable, each clause should be true. Now for each literal(column) j, we remove a snowman or a snowflake based on the value of the literal. i.e if xj is true, we remove all snowflakes from j. Similarly, if xj is false we remove all snowmen from the column.
Hence we are removing either snowmen/snowflakes based on the false value of the literal. Since every variable appears at least in one clause each column contains either a snowman or a snowflake. Also, each clause contains at least one true literal hence no row is empty. We can say that the puzzle is satisfiable.
Conversely, Assume that the puzzle is solvable and consider an arbitrary solution. Consider each column j and assign a value as a literal such that if column j has a snowman, set xj is true and if column j has a snowflake, set xj is false. Otherwise, if column j is empty set xj arbitrarily. Hence, we assign values such that the literals of the remaining entities are all true. Each row has one entity at least, so each clause has at least one true. This makes our final formula ϕ conjunctions of true i.e. true. We conclude that ϕ is satisfiable.

We proved that SAT implies Puzzle and vice versa. And since SAT is NP-hard we can say that solving this is also NP-hard.