1. (*Example for "greedy stays ahead"*) Suppose you are placing sensors on a one-dimensional road. You have identified n possible locations for sensors, at distances $d_1 \leq d_2 \leq ... \leq d_n$ from the start of the road, with $0 \leq d_1 \leq M$ and $d_{i+1} - d_i \leq M$. You must place a sensor within M of the start of the road, and place each sensor after that within M of the previous one. The last sensor must be within M of d_n. Given that, you want to minimize the number of sensors used. The following greedy algorithm, which places each sensor as far as possible from the previous one, will return a list $d_{i_1} \leq d_{i_2} \leq ... \leq d_{i_k}$ of locations where sensors can be placed.

```
GreedySensorMin($d_1...d_n, M$)
  - Initialize an empty list
  - Initialize $I = 1$, $PreviousSensor = 0$.
  - While ($I < n$):
      - While ($I < n$ and $d_{I+1} \leq PreviousSensor + M$) $I++$
      - If ($I < n$) Append $d_I$ to list; $PreviousSensor = d_I$;$I++$.
  - if list is empty, append $d_n$ to list
  - return(list)
```

In using the "greedy stays ahead" proof technique to show that this is optimal, we would compare the greedy solution $d_{g_1}, ..d_{g_k}$ to another solution, $d_{j_1}, ..., d_{j_{k'}}$. We will show that the greedy solution "stays ahead" of the other solution at each step in the following sense:
Claim: For all $t \geq 1, g_t \geq j_t$.

(a) Prove the above claim using induction on the step t. Show base case and induction step.

(b) Use the claim to argue that $k' \geq k$. (Note that this completes the proof of optimality of the greedy algorithm since it shows that greedy algorithm places at most as many sensors as any other solution.)

(c) In big-O notation, how much time does the algorithm as written take? Write a brief explanation.

2. (*Example for "modify the solution"*) You have n cell phone customers who live along a straight highway, at distances $D[1] < D[2] < ... < D[n]$ from the starting point. You need to have a cell tower within Δ distance of each customer, and want to minimize the number of cell towers.

   (*For example, consider* $\Delta = 3$ *and there are* 3 *customers (i.e.,* $n = 3$*) with* $D[1] = 3, D[2] = 7, D[3] = 10$. In this case, you can set up two cell towers, one at 6 and one at 10.)

   Here is a greedy strategy for this problem.

   > **Greedy strategy**: Set up a tower at distance d which is at the farthest edge of the connectivity range for the customer who is closest to the starting point. That is, $d = \Delta + D[1]$. Note that all customers who are within Δ distance of this tower at d, are covered by this tower. Then recursively set up towers for the remaining customers (who are not covered by the first tower).

   We will show that the above greedy strategy gives an optimal solution using modify-the-solution. For this, we will first need to prove the following exchange lemma.

   Exchange Lemma: *Let G denote the greedy solution and let g_1 be the location of the first cell phone tower set up by the greedy algorithm. Let OS denote any solution that does not have cell phone tower at g_1. Then there exists a solution OS' that has a cell phone tower set up at g_1 and OS' has the same number of towers as OS.*

   Proof. Let $OS = \{o_1, ..., o_k\}$. That is, the locations of the cell phone towers as per solution OS is $o_1 < o_2 < ... < o_k$. We ask you to complete the proof of the exchange lemma below.

   (a) Define OS'.
   (b) OS' is a valid solution because ... (*justify why OS' provides coverage to all customers.*)
   (c) The number of cell phone towers in OS' is at most the number of cell phone towers in OS because... (*justify*)

   We will now use the above exchange lemma to argue that the greedy algorithm outputs an optimal solution for any input instance. We will show this using mathematical induction on the input size (i.e., number of customers). The base case for the argument is trivial since for $n = 1$, the greedy algorithm opens a single tower which is indeed optimal.

   (i) Show the inductive step of the argument.

   Having proved the correctness, we now need to give an efficient implementation of the greedy strategy and give time analysis.

   (a) Give an efficient algorithm implementing the above strategy, and give a time analysis for your algorithm.