Problem 1: Runtime Analysis

a. **Θ (log(log(n))**
   - i = 2 : takes Θ(1) time
   - Since i is updated exponentially (i = i * i), after j iterations, i will equal to 2^2^j
   - Loop is done when i == n, therefore, finding k : 2^2^k = n
   - k = log(log_2 (n))
   - Inside the loop, it takes Θ(1) time
   - Θ(log(log_2 (n)) * Θ(1) = Θ(log(log_2 (n))

b. **Θ (n^3)**
   - Outer loop from 1 to n takes Θ(n) time
   - Runs pow (i, 3) when the condition is true (i is a multiple of sqrt(n))
   - Each multiple of sqrt(n), loop runs i^3 times.
   - Leading term is n^3

c. **Θ(n^2 (log(n))**
   - i runs from 1 to n takes Θ(n) time
   - k runs from 1 to n takes Θ(n) time
   - A[k] == i comparison takes Θ(1) time
   - m = 1 <= n, with m being doubled each iteration takes Θ(log(n) time
   - Each iteration does something in Θ(1) time
   - Θ(n) * Θ(n) * Θ(log(n)) = Θ(n^2 (log(n))

d. **Θ (n)**
   - Initializing array takes Θ(1) time
   - When i == size (10), the array must be resized
   - Resizing all iterations takes Θ(n) because the total elements copied is a geometric series
   - a[i] = i * i takes Θ(1) time
   - Θ(n) * Θ(1) = Θ(n)

Problem 2: Linked List Recursion Tracing

a. in1 = 1,2,3,4 and in2 = 5,6
   - Starts with in1 → 1
   - Because in1 is not a null pointer, in1 → next, resulting in a recursive call llrec with in1 → next
   - in → 2
   - Same conditional checks as before, in1 → next results in a recursive call llrec with in1 → 3
   - Same conditional checks as before, in1 → next results in a recursive call llrec with in1 → 4

- As in1 → 4, the recursive call llrec results in in1 → next, where in1 is now pointing to a null pointer
- Because in1 → next is null, function returns in2
- The function returns to goes back to previous recursion call and has 4 → next to 5 → 6
- The function returns the second previous recursion call and sets 3 → next to 4 → 5 → 6
- The function returns the third previous recursion call and sets 2 → next to 3 → 4 → 5 → 6
- The function returns the initial call and sets 1 → next to 2 → 3 → 4 → 5 → 6
- Final linked list is: 1 → 2 → 3 → 4 → 5 → 6

b. in1 = nullptr and in2 = 2?
   - Function checks if ln1 is a null pointer
   - Function returns in2
   - Final linked list: 2