Sample quiz 4 answers.

Smallville is the last city on Earth not saturated by Big Bucks coffee shops. Smallville has one business street with \( n \) blocks. The profit associated with putting a coffee shop on block \( i \) in given in an array \( \text{Profit} \) as \( \text{Profit}[i] \). However, they cannot put coffee shops within \( d \geq 1 \) blocks from each other, i.e., if a shop is in block \( i \) then there cannot be one in blocks \( i - d, i - d + 1, i - 1 \) or \( i + 1, i + 2, \ldots, i + d \).

An backtracking algorithm for computing the maximum total profit of Big-Bucks coffee shops is as follows:

\[
\text{BTBigBucks}(d, \text{Profit}[1..n])
\]

1. IF \( n = 0 \) return 0.
2. IF \( n \leq d + 1 \) return \( \max_{1 \leq i \leq n} \text{Profit}[i] \)
3. \( \text{Case1} \leftarrow \text{Profit}[1] + \text{BTBigBucks}(d, \text{Profit}[d+2..n]) \) {If we put a shop in block 1, we cannot put one in \( 2, \ldots, d + 1 \)}
4. \( \text{Case2} \leftarrow \text{BTBigBucks}(d, \text{Profit}[2..n]) \) {If we don’t put a shop in block 1, there are no other restrictions}
5. Return \( \max(\text{Case1}, \text{Case2}) \)

**Part1: 5 points** Illustrate this algorithm on the following inputs: \( d = 2, n = 8, \text{Profit}[1..8] = 2, 4, 3, 7, 8, 4, 7, 5 \) (as a tree of recursive calls and answers).

We first branch on whether to build a store in the first block. If we do, we get 2 + a recursive solution to 7,8,4,7,5. If we don’t, we get a recursive solution to 4,3,7,8,4,7,5.

In the first case, we choose between 7+ BigBucks((7,5), 2) and BigBucks((8,4,7,5), 2). Since the first part has \( n = 2 \leq 2 = d \), we immediately return \( \max(7) = 7 \), which makes a total of 14. The second part, we choose between 8 +BigBucks((5),2)= 8+5 =13 and BigBucks(4,7,5)= max(4+ BigBucks(empty, 2)=4+0, BigBucks(7,5)=max(7,5)=7)=7. Thus, the second case return 13. Since this is less than 14, the first case is 14.

In the second case, we choose between 4+BigBucks((8,4,7,5), 2) and BigBucks((3,7,8,4,7,5), 2) We already simulated the algorithm on (8,4,7,5) above to get 13. The other option chooses between 3+BigBucks((4,7,5), 2) and BigBucks((7,8,4,7,5),2). We already simulated the algorithm on (4,7,5) and got 7, and on (7,8,4,7,5) and got 14. Thus, the max for the second option is 14, which is less than 4+13. So we would return 17 for the second case, which is greater than 14. So the final answer returned is 17, the profit from picking the second block, the fifth block and the last block.

**Part2: 5 points** Give an upper bound on the number of recursive calls the above algorithm makes, in the worst-case. (Some points will be based on how tight the bound is. Be sure to explain your answer.)
The algorithm makes one recursive call to an input of size $n - 1$ and the other to an input of size $n - d - 1$. Since $d \geq 1$, the second is of size at most $n - 2$. Thus, $T(n) \leq T(n - 1) + T(n - 2) + O(1)$, which grows at the same order as the $n$th Fibonacci numbers, $O((1 + \sqrt{5}/2)^n)$ which is about $2^{\sqrt{5}n}$.

**Part 3: 10 points** Give a dynamic programming version of the recurrence.

The algorithm doesn’t change $d$ in recursive calls, and only deletes initial segments of the array $A$. So the recursive calls are all of the form $BigBucks(A[I..n], d)$ for $1 \leq I \leq n$ (or the empty array, which we view as $I = n + 1$). The matrix $B[I]$ will compute and store the values $BigBucks(A[I..n], d)$ using the same recurrence.

$$\text{DPBigBucks}(A[1..n], d)$$

1. Initialize $B[1..n + 1]$
2. $B[n + 1] \leftarrow 0.$
3. FOR $I = n$ downto $n - d$ do:
   4. $B[I] \leftarrow \max(A[I], B[I + 1])$
5. FOR $I = n - d - 1$ downto 1 do:
   6. $B[I] \leftarrow \max(A[I] + B[I + d + 1], B[I + 1])$
7. Return $B[1]$

**Part 4: 5 points** Give a time analysis of this dynamic programming algorithm.

The algorithm has two loops, one of $O(d) \leq O(n)$ and the other of $O(n)$. So the total time is $O(n)$.

**Part 5: 5 points** Show the array that your dynamic programming algorithm produces on the above example. $d = 2, n = 8, A[1..8] = 2, 4, 3, 7, 8, 4, 7, 5$