Consider the problem specification below:
Problem statement: Find all pairs of numbers (x, y) in an array of numbers such that x+y≥K, a given number.

Well-defined specification:
Input: an array A [p...r] of n **unordered** numbers, **n≥0**, and K, a given number.
Output: zero or more pairs of numbers (x, y)
Correctness criteria:
(x, y) is in the output **if and only if** x is in A, y is in A, x+y≥K, x and y are in different array cells, and x appears before y in A.

An Example: If A[1...5] = [1, 2, 0, 5, 5, 6] and K = 5, output should be the pairs (1, 5), (1, 5), (1, 6), (2,5), (2, 5), (2, 6), (0, 5), (0, 5), (0, 6), (5, 5), (5, 6) and (5,6)

1. The inherent complexity of the above problem is:
A. $\Omega(n^3)$ B. O(1) C. $\Omega(n)$ D. O(n) E. $\Omega(n^2)$

2. Four iterative strategies are given below to solve this problem: Suppose "(i) If A is empty or has only one number quit without outputting anything" is a common first step of all four. Remaining steps are given below for each strategy.

Strategy I. (ii) Check the sum of all pairs of numbers in A in which the first number in the pair is A[p] and the second number is any other number in A to the right of A[p] and output those pairs that sum to a value ≥ K; (iii) then do the same with the first number being A[p+1] and the second number is any other number in A to the right of A[p+1]and output those pairs whose sum ≥ K, then consider A[p+2] as the first number of the pair, etc. until the last pair considered is (A[r–1], A[r]).

Strategy II. (ii) Scan A to locate all numbers in A ≥ K and place those in a list; (iii) Then output all pairs of numbers in A in which the first number in the pair is the first number from the previous list and the second number is any other number in A that appears to the right of this first number; (iv) repeat this for each of the other numbers in the list from (ii).

Strategy III. (ii) Output (A[p], A[p+1]) if their sum ≥ K; (iii) do the same for pairs (A[p+1], A[p+2]), (A[p+2], A[p+3]), (A[p+3], A[p+4]), and so on until the sum of the last pair (A[r-1], A[r]) is considered.

Strategy IV. (ii) Consider A[p...r] and the first number in A, A[p], and compute *difference* = K–A[p]; (iii) output all pairs (A[p], y) where y is any number in the subarray A[p+1...r] such that y ≥ *difference*; (iv) repeat (ii) and (iii) for the subarray A[p+1...r] – i.e., consider A[p+1...r] and its first number, A[p+1], and compute *difference* = K–A[p+1] and output all pairs (A[p+1], y) where y is any number in the subarray A[p+2...r] such that y ≥ *difference* – then repeat (ii) and (iii) for the subarray A[p+2...r], and so on until the last subarray considered is A[r-1...r].

A. None of these strategies are correct.
B. All of these strategies are correct.
C. Exactly one of these strategies is correct.
D. Exactly two of these strategies are correct.
E. Exactly three of these strategies are correct.

3. Consider the following correct recursive strategy to solve this problem. Some parts of it are missing. Select the correct answer to complete the missing part.
(i) If A[p...r] is empty or has only one number quit without outputting anything
(ii) Otherwise ____
(iii) Recursively solve the problem for the subarray A[p+1...r]

A. (ii-a) compute *difference* = K–A[p];
(ii-b) output all pairs (A[p], y) where y is a number in the subarray A[p+1...r] and y ≥ *difference*;
(ii-c) compute *difference* = K–A[p+1];
(ii-d) output all pairs (A[p+1], y) where y is a number in the subarray A[p+2...r] & y ≥ *difference*;
(ii-e) repeat for A[p+2] and subarray A[p+3...r], and so on until and including for A[r–1] and subarray A[r...r].

B. (ii-a) compute *difference* = K–A[p];
(ii-b) output all pairs (A[p], y) where y is a number in the subarray A[p+1...r] and y ≥ *difference*.

C. (ii-a) output all pairs (A[p], y) where y is a number in the subarray A[p+1...r] and y ≥ K;
(ii-b) output all pairs (A[p+1], y) where y is a number in the subarray A[p+2...r] and y ≥ K;
(ii-c) repeat for A[p+2] and subarray A[p+3...r], and so on until and including for A[r–1] and subarray A[r...r].

D. (ii-a) output all pairs (A[p], y) where y is a number in the subarray A[p+1...r] and y ≥ K;

E. None of the above is the correct way to complete the missing part of the recursive strategy.

4. Which of the following is a correct recursive divide & conquer strategy to solve this problem?

A. (i) If A[p...r] is empty or has only one number quit without outputting anything
(ii) Otherwise recursively solve the problem for the left and right halves of the array.

B. (i) If A[p...r] is empty or has only one number quit without outputting anything
(ii) Otherwise recursively solve the problem for the left and right halves of the array.
(iii) output all pairs (x, y) where x is a number from the left half and y is a number from the right half.

C. (i) If A[p...r] is empty or has only one number quit without outputting anything
(ii) Otherwise recursively solve the problem for the left and right halves of the array.
(iii) output all pairs (x, y) where x is a number from the left half and y is a number from the right half such that x+y≥K.

D. (i) If A[p...r] is empty or has only one number quit without outputting anything
(ii) Otherwise recursively solve the problem for the left and right halves of the array.
(iii) output all pairs (x, y) where x is a number from the left half, y is a number from the right half, and either x or y is ≥ K.

E. None of the above is a correct recursive divide & conquer strategy.

5. Are other correct computational strategies possible to solve this problem? Choose one:

yes (mark T on scantron)   no (mark F on scantron)

Consider the computational strategy of checking <u>every pair</u> of numbers (x, y) in A such that x and y are in different array cells and x appears before y in A to see if their sum ≥ K, and outputting only those pairs that meet this criterion. To estimate the complexity of this strategy, one needs to come up with an <u>estimate of how many pairs need to be considered in total</u>. Suppose the numbers in A are: [a_1, a_2, a_3,...,a_{n-1}, a_n]

6. How many pairs in which the first number is $\mathbf{a_1}$ should be looked at?

A. n   B. n–1   C. n–2   D. n–3   E. n+1

7. How many pairs in which the first number is $\mathbf{a_2}$ should be looked at?

A. n   B. n–1   C. n–2   D. n–3   E. n+1

8. What is the mathematical expression for the total number of pairs that need to be looked at?

A. n+1   B. n^2 C. n–1   D. 1+2+3+...+n

E. None of these is the mathematical expression for the total number of pairs that need to be looked at

9. What is the correct summation for the total number of pairs that need to be looked at?

A. $\sum_{i=1}^{i=n} i$ B. $\sum_{i=1}^{i=n} n$ C. $\sum_{i=1}^{i=n-1} n$ D. $\sum_{i=1}^{i=n-1} i$

E. None of these is the correct summation for the total number of pairs that need to be looked at

10. What is the exact mathematical expression for the total number of pairs that need to be looked at?

A. n(n–1)/2   B. n(n–1)   C. n(n+1)/2   D. n(n+1)   E. none of these

Hint: The sum of first integers = 1+2+...+n = $\sum_{i=1}^{i=n} i$ = n(n+1)/2

Consider the following recursive strategy to solve this problem and the algorithm implementing this strategy. Some parts of the algorithm are missing. Select the correct answers for the next five questions to complete all the missing parts.

(i) If A[p...r] is empty or has only one number quit without outputting anything

(ii) Otherwise output all pairs (A[p], y) where y is a number in subarray A[p+1...r] such that A[p]+y≥K

(iii) Recursively solve the problem for the subarray A[p+1...r]

```
pairs-recursive(A[p...r]: array of number, K: number)
if p<r then
        for i=p+1 to r
                if A[p]+A[i]≥K then
                        output (A[p], A[i])
        pairs-recursive(A[p+1...r], K)
```

11. What is the correct answer to fill the blank in the algorithm marked Q.11?

A. p>r   B. p!=r   C. p≥r   D. p≤r   E. p<r

12. What is the correct answer to fill the blank in the algorithm marked Q.12?

A. i=p to r   B. i=p to r–1   C. i=p+1 to r   D. i= p+1 to r–1   E. none of these

13. What is the correct answer to fill the blank in the algorithm marked Q.13?
A. A[p]+A[i]≥K B. A[p]+A[i]>K C. A[p]+A[p+1]≥K D. A[p]+A[p+1]>K E. none of these

14. What is the correct answer to fill the blank in the algorithm marked Q.14?
A. (A[p], A[p+1]) B. (A[p], A[i]) C. (A[p], A[r]) D. (A[p], A[p]) E. (A[i], A[i])

15. What is the correct answer to fill the blank in the algorithm marked Q.15?
A. (A[p...r–1], K) B. (A[p...r], K) C. (A[p+1...r], K) D. (A[p+1... r–1], K) E. none of these

Consider the following recursive divide & conquer strategy to solve this problem and the algorithm implementing this strategy. Some parts of the algorithm are missing. Select the correct answers for the next five questions to complete all the missing parts.
(i) If A[p...r] is empty or has only one number quit without outputting anything.
(ii) Otherwise recursively solve the problem for the left and right halves of the array.
(iii) Then consider all pairs (x, y) with x drawn from the left half and y drawn from the right half and output those whose sum is greater than K.

```
pairs-recursive-divide-conquer(A[p...r]: array of number, K: number)
1 if r–p>0then
2.      m=floor((p+r)/2)
3.      pairs-recursive-divide-conquer(A[p...m], K)          //two choices for lines 3 & 4
4.      pairs-recursive-divide-conquer(A[m+1...r], K)
3.      pairs-recursive-divide-conquer(A[m+1...r], K)
4.      pairs-recursive-divide-conquer(A[p...m], K)
5.      for i=p to m
6.          for j=m+1 to r
7.              if A[i]+A[j]≥K then output (A[i], A[j])
```

16. What is the correct answer to fill the blank in the algorithm marked Q.16 so that m points to the index of the middle number in A?
A. floor((p+r)/2) B. floor((r–p)/2) C. ceiling((r–p)/2) D. n/2 E. none of these

17. What is the correct answer to fill the blank in the algorithm marked Q.17?
A. (A[p...r], K) B. (A[p...m], K) C. (A[m+1...r], K) D. (A[m...r], K) E. (A[p...m+1], K)

18. What is the correct answer to fill the blank in the algorithm marked Q.18?
A. (A[p...r], K) B. (A[p...m], K) C. (A[m+1...r], K) D. (A[m...r], K) E. (A[p...m+1], K)

17. What is the correct answer to fill the blank in the algorithm marked Q.17?
A. (A[p...r], K) B. (A[p...m], K) C. (A[m+1...r], K) D. (A[m...r], K) E. (A[p...m+1], K)

18. What is the correct answer to fill the blank in the algorithm marked Q.18?
A. (A[p...r], K) B. (A[p...m], K) C. (A[m+1...r], K) D. (A[m...r], K) E. (A[p...m+1], K)

19. What is the correct answer to fill the blank in the algorithm marked Q.19?
A. p to r B. p to m+1 C. m to r D. p to m E. m+1 to r

20. What is the correct answer to fill the blank in the algorithm marked Q.20?
A. p to r B. p to m+1 C. m to r D. p to m E. m+1 to r

```
Mystery(n: non-negative integer)
    if n ≤ 1 then return 0
    else return (1+Mystery(n-2))
```

21. True or False? Mystery is a tail recursive algorithm.

22. Which of the following is correct?
A. Mystery is an example of iteration.
B. Mystery is an example of general tail recursion.
C. Mystery is an example of strict tail recursion.
D. Mystery is an example of a recursive divide & conquer algorithm.
E. None of these statements are correct.

23. How many nodes, including the root node for the original execution and any leaf nodes for base case executions, does the Recursion Tree of Mystery(5) have?
A. 1 B. 2 C. 3 D. 4 E. 5

24. True or False? Mystery suffers from duplicated work.

25. What is the function of n Mystery computes?
A. floor(n/2) B. ceiling(n/2) C. floor((n–2)/2) D. ceiling((n–2)/2) E. none of these