Name: ______________________________

1. (1 pt.)
   - **Read all material carefully.**
   - *If in doubt whether something is allowed, ask, don't assume.*
   - You may refer to your books, papers, and notes during this test.
   - E-books may be used *subject to the restrictions* noted in class.
   - Computers are not permitted, except when used strictly as ebooks.
   - Network access of any kind (cell, voice, text, data, ...) is not permitted.
   - Write, and draw, carefully. Ambiguous or cryptic answers receive zero credit.
   - Use class and textbook conventions for notation, algorithmic options, etc.

   Write your name in the space provided above.

WAIT UNTIL INSTRUCTED TO CONTINUE TO REMAINING QUESTIONS.

Do not write in the following table.

Q	Full	Score
1	1	
2	9	
3	10	
4	15	
5	15	
6	10	
total	60	

2. (9 pts.) For the following mapping of rod lengths to prices, how many recursive invocations of CUT-ROD does the *recursive top-down cut-rod algorithm* make, when invoked with the following array p and $n = 12$? *Provide an exact numerical answer along with an explanation.* [Hint: You do not need to solve the cut-rod instance.]

length:	1	2	3	4	5	6	7	8	9	10	11	12
price:	4	7	9	14	18	22	30	30	28	38	40	44

3. (10 pts.) Solve the following recurrences. *Clearly state the methods you use for your solutions and outline their key steps.* (Show your work.)

   (a) $T(n) = 2T(n/2) + 13n + 3$

   (b) $S(n) = 7S(n/2) + 8n^{1.75}$

[additional space for answering the earlier question]

4. (15 pts.) Trace the operation of the LCS-Length algorithm on the following sequences.

```
A C B A A B A
C B A C A A B
```

Depict the state of the b and c arrays (1) after four iterations of the outer nested loop and (2) at the end of the algorithm.

[additional space for answering the earlier question]

5. (15 pts.) Consider the following Java fragment from a recent class exercise:

```
public static int search(int[] haystack, int needle) {
    int lo = 0;
    int hi = haystack.length - 1;
    while(lo + 1 < hi) {
        int mid = (lo + hi) / 2;
        if(haystack[mid] > needle) hi = mid;
        else if (haystack[mid] < needle) lo = mid;
        else return mid;
    }
    for(int i = lo; i <= hi; i++) {
            if(haystack[i] == needle) return i;
    }
    return -1;
}
```

(a) State a recurrence equation for $T(n)$, the running time of the above code as a function of n, the length of the `haystack` array.

(b) Explain why the above recurrence is correct.

(c) Solve the recurrence using one of the methods in the textbook. (State the method and show its key steps.)

[additional space for answering the earlier question]

6. (10 pts.) Depict the *first three levels* of the recursion tree that outlines the recursive calls made by the FIND-MAXIMUM-SUBARRAY algorithm when invoked on the following array, with `low` and `high` equal to 1 and 10, respectively.

   The *nodes* of the tree should be labeled with the function invoked: FIND-MAXIMUM-SUBARRAY (M) or FIND-MAX-CROSSING-SUBARRAY (X).

   The *edges* should connect each function's node (child) to the node of its invoker (parent).

i:	1	2	3	4	5	6	7	8	9	10
A[i]:	88	-1	-11	-23	43	-6	8	-19	-58	50

[additional space for answering the earlier question]