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.
   - No computer or network access of any kind is allowed (or needed).
   - Write, and draw, carefully. Ambiguous or cryptic answers receive zero credit.
   - Use class and textbook conventions for notation, algorithmic options, etc.
   - Budget your time: roughly one minute per point.

   Write your name in the space provided above.

2. (14 pts.) Solve the rod-cutting problem for a rod of length 12 and the following price table. Use the EXTENDED-BOTTOM-UP-CUT-ROD algorithm. In particular:

   (a) Depict the final states of the `r` and `s` arrays used by that algorithm.

   (b) Indicate the distances of the optimal cuts from the left end of the original rod.

length:	1	2	3	4	5	6	7	8	9	10	11	12
price:	7	8	18	16	30	12	28	24	72	40	33	48

[additional space for answering the earlier question]

3. (5 pts.) Solve the following recurrence. *Clearly state the method you use for your solution and outline its key steps.* (Show your work.)

$$T(n) = 6T(n/2) + 3n \log n + 17$$

4. (10 pts.) Trace the operation of the LCS-Length algorithm on the following sequences.

```
C A B A A C B A
C A A C A B 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. (10 pts.) Trace the operation of the PRINT-LCS algorithm on the result of Question 4. Provide the arguments for each of recursive call of PRINT-LCS.

6. (10 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]

7. (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]`:	4	4	3	-2	3	-1	-2	-1	2	4

[additional space for answering the earlier question]