Questions and Exercises to work out and turn in:

Grading Guidelines:

* A right answer will get full credit when:

1. It is right (worth 25%)
2. It is right **AND** neatly presented making it easy and pleasant to read. (worth an **extra** 15%)
3. There is an **obvious and clear link** between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth an **extra** 60%).
4. Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.

You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, **personal** writing is expected.

* USE THIS FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **DO NOT DELETE ANYTHING FROM THIS FILE:** JUST **INSERT** YOUR ANSWERS.
* IF USING HAND WRITING (STRONGLY DISCOURAGED), **USE THIS FILE** BY CREATING SUFFICIENT SPACE AND WRITE IN YOUR ANSWERS.
* FAILING TO FOLLOW TURN IN DIRECTIONS /GUIDELINES WILL COST **A 30% PENALTY.**

Objectives of this assignment:

* to use and manipulate the concepts presented in this module
* to propose and write algorithms in pseudocode
* to analyze the time complexity of algorithms
* to analyze the space complexity of algorithms
* to learn autonomously new concepts

What you need to do:

Answer the questions and/or solve the exercises described below.

Exercise 1 (50 points)

Consider a modification of the rod-cutting problem in which, in addition to a price pi for each rod, each cut incurs a fixed cost of c. The revenue associated with a solution is now the sum of the prices of the pieces minus the costs of making the cuts. Give a dynamic-programming algorithm to solve this modified problem. Precisely comment and explain the modications/additions you make to the original program to meet the requirements.

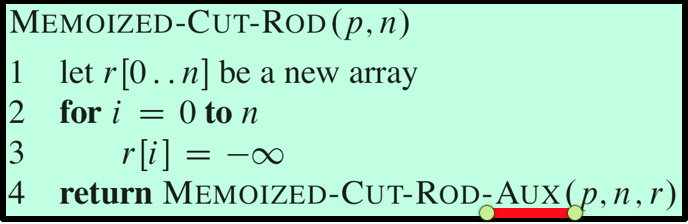
EXTENDED-BOTTOM-UP-CUT-ROD-WITH-COST(p, n, c)

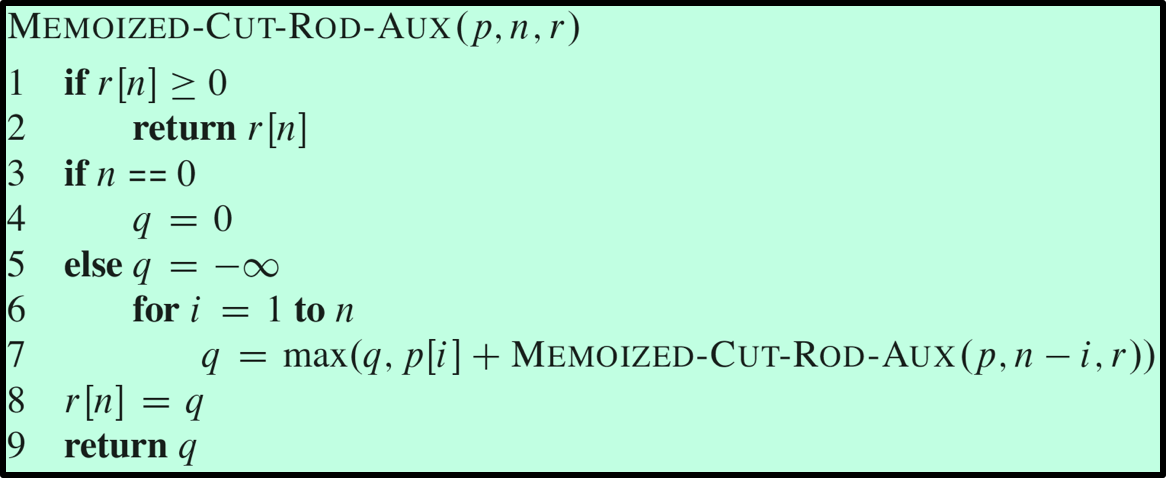
1. // Added c to the input so that the user can input the cost of each cut
2. Let r[0…n] and s[1…n] be new arrays
3. r[0] = 0
4. for j = 1 to n
5. q = -∞
6. for i = 1 to j
7. // Adding an if statement that determines whether a cut will be made. If so, the cost c will be taken into account.
8. if i < j
9. if q < p[i] + r[j – i] - c
10. q = p[i] + r[j – i] - c
11. s[j] = i
12. // Adding an else statement that does not take the cost into account because no cuts were made.
13. else
14. if q < p[i] + r[j – i]
15. q = p[i] + r[j – i]
16. s[j] = i
17. r[j] = q
18. return r and s

This new algorithm has a few changes to it in order to incorporate a cut-cost c. The algorithm has to know what the value of c is, so in the input I added a parameter c for the user to input the cost of 1 cut. Then, I have to incorporate c into the algorithm whenever a cut is made. In order to find out if a cut is made, I used the fact that j is the value of the length of the rod, and i is the number of inches of where the rod should be cut. That means if i < j, the rod is being cut, so I added an if statement to determine if that is the case. In that case, the cost should be incorporated by calculating the value of the cut rods put together in the same way the original algorithm does it, minus the cost of a cut. Since the cost of a cut is incorporated for each length of the rod, I only need to account for 1 extra cut, not the total number of cuts. Finally, I needed to add an else statement for when i is not < j, namely when i == j. When i == j, there is no cut, so the cost of a cut must not be incorporated. That is why c does not appear in the else statement in the algorithm above.

Exercise 2 (50 points)

1. **Implement** in your preferred language MEMOIZED-CUT-ROD. Insure your program can be compiled and executed on an Engineering Unix Tux machine.
2. Test your implementation that it yields the same results as in the textbook. State here whether your implements yields the same results as in the textbook.
3. Modify your MEMOIZED-CUT-ROD implementation (fromQuestion 1)to return not only the value but the actual solution, too.
4. In addition to the pseudocode in THIS file, turn in the source code of your implementation. Include a small report stating 1) whether your code works, 2) how to compile and execute your program on a Tux machine.





What you need to turn in:

* Electronic copy of this file (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.
* Recall that answers must be well written, documented, justified, and presented to get full credit.
* How this assignment will be graded:
* A right answer will get full credit when:
* It is right (worth 25%)
* It is right AND neatly presented making it easy and pleasant to read. (worth 15%)
* There is an obvious and clear link between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth 60%).
* Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.
* You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, personal writing is expected.