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** EACH ANSWER **RIGHT AFTER ITS QUESTION/PROMPT**.

- 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 (30 points)

Consider a modification of the rod-cutting problem in which, in addition to a price p_i 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 modifications/additions you make to the original algorithm to meet the requirements.

Include here the pseudocode
Make sure you comment, justify, and explain the modifications/additions.
Cite your references.

```
cutRod(p, n, c)
  // Array to store max revenue for lengths 0 to n
   let r[0..n] be a new array
  // Initialize the arrat to the min value to show uncalculated revenues
   for i = 0 to n
      r[i] = -∞
   return cutRodHelper(p, n, c, r)

cutRodHelper(p, n, c, r)
   // Return memorized result if already calculated
   if r[n] >= 0
      return r[n]
   // Base Case: Return 0 for a length of 0
   if n == 0
      return 0
   else
      // Reset q to min value for each call
      let q = -∞
      for i = 1 to n
         if i < n
            revenue = p[i] + cutRodHelper(p, n-i, c, r) - c
         else
            revenue = p[i] + cutRodHelper(p, n-i, c, r)
         // Update max revenue if larger than current max
         q = MAX(q, revenue)
     // Memoize the result for the current rod length
      r[n] = q
      return q

main
   let p = [0, 1, 5, 8, 9, 10, 17, 17, 20, 24, 30] // prices for lengths 1 to n
   let n = 10 // total length of the rod
   let c = 1 // cost per cut
   for i = 0 to n
      print "Maximum revenue for length", i, ":", cutRod(p, i, c)
```

In the updated code, I added a new variable named c to keep track of the cost for each cut we make when we're trying to figure out how to get the most money from cutting rods. I thought setting the cost to 1 would make sense—it's a small cost but still affects the final amount we can earn from cutting. Because of this new cost thing, I had to change the cutRod and cutRodHelper functions a bit so they can deal with this cost.

Also, I put in an if-else statement to handle the cost better. Basically, if we're actually cutting the rod (which is when i, the size of the cut we might make, is less than n, the whole rod's length), we take the cost off the money we could make. This part makes sure we're being realistic about what it costs to cut the rod. But if we don't need to cut the rod, the else part makes sure we're not taking off any cost, which makes sense because we're not actually making a cut. This tweak helps us get a more accurate idea of the money we could make, taking the cutting cost into account but not overdoing it.

The only references I used for creating this modification was the lectures and chapter 15.1 in the text book.

Exercise 2 (70 points)

1) (10 points if correct) **Implement** in your preferred language MEMOIZED-CUT-ROD. Insure your program can be compiled and executed on an Engineering Unix Tux machine.

Turn in this program separately on Canvas with your homework

2) (20 points) Test your implementation that it yields the same results as in the textbook. State here whether your implementation yields the same results (max revenue) as in the textbook.

Insert **here** screenshots to show that your program yields the same results.

The screenshots must include the Tux machine name, your Auburn username and the date. For the date, type the command "date" just before executing your program. Your screenshot(s) must be as readable as this template screenshot:

```
bar0086@tux051:~$ date
Tue Apr  9 23:16:53 CDT 2024
bar0086@tux051:~$ javac RodCutting.java
bar0086@tux051:~$ java RodCutting
Maximum revenue: 0
Maximum revenue: 1
Maximum revenue: 5
Maximum revenue: 8
Maximum revenue: 10
Maximum revenue: 13
Maximum revenue: 17
Maximum revenue: 18
Maximum revenue: 22
Maximum revenue: 25
Maximum revenue: 30
bar0086@tux051:~$
```

3) (35 points) Modify your MEMOIZED-CUT-ROD implementation (from Question 1) to return not only the value but the actual solution (how to cut), too.

Include/Insert here the pseudocode of your program

```
cutRod(p, n)
    let memo[0..n] be a new array
    let cuts[0..n] be a new array to track the cuts leading to maximum revenue
    for i = 0 to n
        memo[i] = -∞ // Initialize memoization array with negative infinity

    let maxRevenue = cutRodHelper(p, n, memo, cuts)

    if maxRevenue > p[n] // Check if making cuts yields more revenue than not making any cuts
        print "Cuts made to produce lengths:"
        while n > 0
            print cuts[n], " "
            n -= cuts[n] // Deduct the length of the cut from n to find the next cut
    else
        print "No cuts made"
    return maxRevenue

cutRodHelper(p, n, memo, cuts)
    if n == 0
        return 0 // Base case: No revenue from a rod of length 0
    if memo[n] != -∞
        return memo[n] // Return memoized result if already calculated
```

```
    let maxRevenue = -∞
    for i = 1 to n
        let revenue = p[i] + cutRodHelper(p, n - i, memo, cuts) // Calculate revenue for this cut
        if revenue > maxRevenue
            maxRevenue = revenue
            cuts[n] = i // Record this cut as it leads to max revenue

    memo[n] = maxRevenue // Memoize the maximum revenue for this length
    return maxRevenue

main
    let p = [0, 1, 5, 8, 9, 10, 17, 17, 20, 24, 30] // Prices for lengths 1 to n
    let n = 9 // Total length of the rod to be cut
    for i = 0 to n
        let maxRevenue = cutRod(p, i)
        print "and Maximum revenue for length", i, ":", maxRevenue
```

Turn in the modified implementation separately on Canvas.
Insert **here** screenshots to show that your program yields correct results for the example on the textbook.
The screenshots must include the Tux machine name, your Auburn username and the date. For the date, type the command "date" just before executing your program.

```
bar0086@tux051:~$ date
Tue Apr  9 23:18:52 CDT 2024
bar0086@tux051:~$ javac  RodCuttingCutTracking.java
bar0086@tux051:~$ java  RodCuttingCutTracking
No cuts made and Maximum revenue: 0
No cuts made and Maximum revenue: 1
No cuts made and Maximum revenue: 5
No cuts made and Maximum revenue: 8
Cuts made to produce lengths: 2 2 and Maximum revenue: 10
Cuts made to produce lengths: 2 3 and Maximum revenue: 13
No cuts made and Maximum revenue: 17
Cuts made to produce lengths: 1 6 and Maximum revenue: 18
Cuts made to produce lengths: 2 6 and Maximum revenue: 22
Cuts made to produce lengths: 3 6 and Maximum revenue: 25
No cuts made and Maximum revenue: 30
bar0086@tux051:~$
```

4) (5 points) 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.

include here how to compile and execute your two programs on Tux machines

1) Yes, both programs work.
2) Once logged into the sftp server, navigate to the directory that holds your program on the local machine and use “put” followed by the program name to transfer the program file to the server, then in another window connect to the server, log in and get assigned your random gate to access the files that were just uploaded. Lastly, the programs are compiled by entering “javac” followed by the name of the file you want to compile. The program is then executed by enter “java” followed by the name of the file but leaving off the .java The output is then displayed.

```
MEMOIZED-CUT-ROD(p, n)
1  let r[0..n] be a new array
2  for i = 0 to n
3      r[i] = -∞
4  return MEMOIZED-CUT-ROD-AUX(p, n, r)
```

```
MEMOIZED-CUT-ROD-AUX(p, n, r)
1  if r[n] ≥ 0
2      return r[n]
3  if n == 0
4      q = 0
5  else q = −∞
6      for i = 1 to n
7          q = max(q, p[i] + MEMOIZED-CUT-ROD-AUX(p, n − i, r))
8  r[n] = q
9  return q
```

AUBURN UNIVERSITY

3280 M5-Homework

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.
- Source code of required program(s) with directions about how to compile and execute them.
- 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.

Appendix: Grading: What is an OBVIOUS and CLEAR LINK?

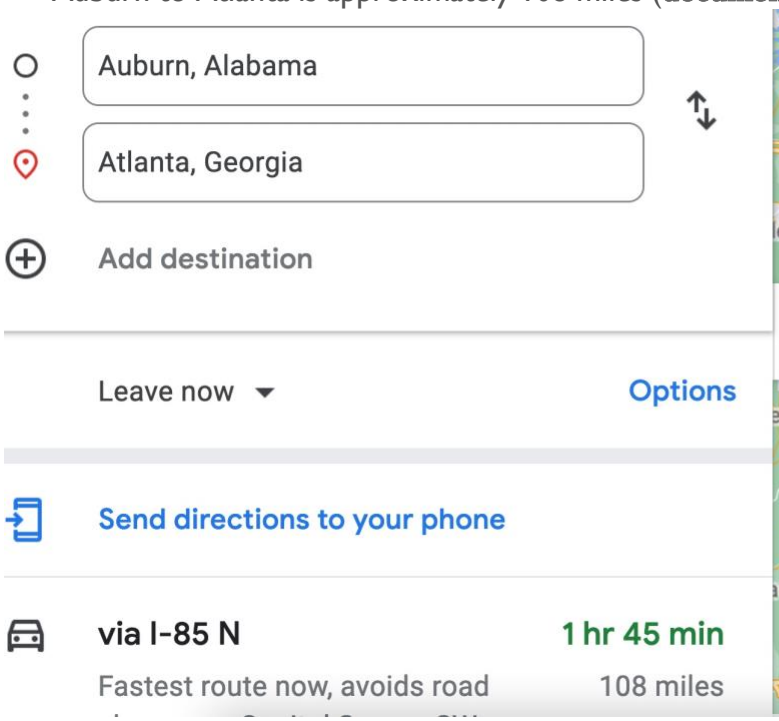
Here is an example to explain what an **obvious and clear** link is and how we grade your work.

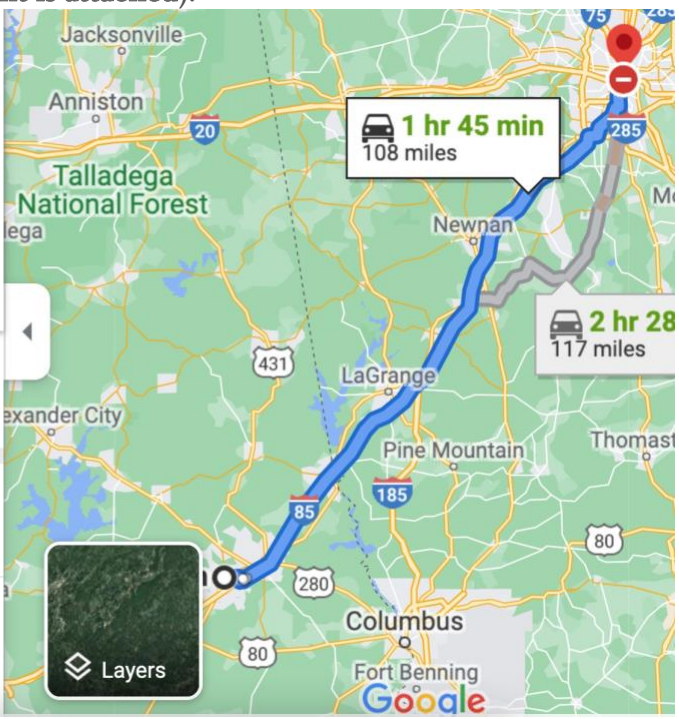
Consider the following problem:

"(100 points) John travels from Auburn to Atlanta in his car at a speed of 60 mph. Leaving at 8am, at what time will John reach Atlanta".

Here are the answers of three students and their scores:

- Student 1 answers: "9:48am". Student 1 will get 25 points.
- Student 2 answers: "John will reach Atlanta at 9:48am". Student 2 will get 25+15 = 40 points
- Student 3 answers: "The time t to travel a distance d at speed v is equal to d/v = d/60mph. The problem does not provide the distance d from Auburn to Atlanta. Based on GoogleMaps, the distance from Auburn to Atlanta is approximately 108 miles (document is attached).





Therefore, the time t = 108 miles/60mph * 60 minutes/hour= 108 minutes. Since John left at 8am, he will then reach Atlanta at 8am + 108 minutes = 8 am + 60 minutes + 48 minutes = 9:48".

Student 3 will get 25 + 15 + 60 = 100 points

Do you see the direct link going from the data provided in the question to the final answer, using general knowledge/formula and documents?.... Can you now solve the following problem and get 100 points?

"(100 points) Alice travels from Auburn to Atlanta in her car at a speed of 60 mph. Leaving at 8am, at what time will Alice reach Atlanta assuming that she had a flat tire that delayed her 30 minutes".