Problem 1: *(2 points)* **Rod Cutting**: (from the text CLRS) 15.1-2

Problem 2: *(3 points)* **Modified Rod Cutting**: (from the text CLRS) 15.1-3

Problem 3: *(10 points)* **0-1 Knapsack**: **Recursive vs DP**

Given weights and values of n items, select items to place in a knapsack of capacity W to maximize the total value in the knapsack. That is, given two integer arrays val[] and wt[ ] which represent values and weights associated with n items respectively and an integer W which represents knapsack capacity, determine the maximum value subset of val[] such that sum of the weights of this subset is ≤ W. Items cannot be broken or used more than once, you either select the complete item, or don't select it. Implement both a recursive and dynamic programming algorithm to solve the 0-1 knapsack problem. Both algorithms should return the maximum total value of items that can fit in the knapsack.

   a) Give a verbal description and detailed pseudo-code for each algorithm.

   b) Implement both algorithms in one program named knapsack. Your program should randomly generate test cases that are solved using both the DP and Recursive algorithm. The program should output to the terminal: n, W, time for the DP algorithm, max for the DP, time for the Recursive algorithm, max for Recursive.   The max values should be the same. Sample output is below

```
N=10 W =100   Rec time = 0.0122 DP time = 0.0103 max Rec = 478 max DP = 478
N=15 W =100   Rec time = 0.1752 DP time = 0.0145 max Rec = 676 max DP = 676
N=20 W =100   Rec time = 2.4546 DP time = 0.0201 max Rec = 739 max DP = 739
N=25 W =100   Rec time = 16.0216 DP time = 0.0279 max Rec = 617 max DP = 617
N=30 W =100   Rec time = 192.209 DP time = 0.0323 max Rec = 721 max DP = 721
N=35 W =100   Rec time = 1891.7 DP time = 0.0354 max Rec = 970 max DP = 970
N=40 W =100   Rec time = 3782.96 DP time = 0.0416 max Rec = 740 max DP = 740
```

   c) Collect running times for randomly generated input, plot the data and fit a curve. If the recursive algorithm is too slow you can collect data using different values of W and n.

   d) Discuss your implementation, results and how you collected the data. How does W change the running time?

Problem 4: **Shopping Spree:** *(15 points)* Acme Super Store is having a contest to give away shopping sprees to lucky families. If a family wins a shopping spree each person in the family can take any items in the store that he or she can carry out, however each person can only take one of each type of item. For example, one family member can take one television, one watch and one toaster, while another family member can take one television, one camera and one pair of shoes.   Each item has a price (in dollars) and a weight (in pounds) and each person in the family has a limit in the total weight they can carry. Two people cannot work together to carry an item. Your job is to help the families select items for each person to carry to maximize the total price of all items the family takes. Write an algorithm to determine the maximum total price of items for each family and the items that each family member should select.

Submit to Canvas

a) A verbal description and give pseudo-code for your algorithm. Try to create an algorithm that is efficient in both time and storage requirements.

b) What is the theoretical running time of your algorithm for one test case given N items, a family of size F, and family members who can carry at most M_i pounds for $1 \leq i \leq F$.

c) Implement your algorithm by writing a program named "**shopping**" (in C, C++ or Python) that compiles and runs on the OSU engineering servers. The program should satisfy the specifications below.

Input: The input file named "**shopping.txt**" consists of T test cases

- T $(1 \leq T \leq 100)$ is given on the first line of the input file.
- Each test case begins with a line containing a single integer number N that indicates the number of items $(1 \leq N \leq 100)$ in that test case
- Followed by N lines, each containing two integers: P and W. The first integer $(1 \leq P \leq 5000)$ corresponds to the price of object and the second integer $(1 \leq W \leq 100)$ corresponds to the weight of object.
- The next line contains one integer $(1 \leq F \leq 30)$ which is the number of people in that family.
- The next F lines contains the maximum weight $(1 \leq M \leq 200)$ that can be carried by the ith person in the family $(1 \leq i \leq F)$.

Output: Written to a file named "**results.txt**". For each test case your program should output the maximum total price of all goods that the family can carry out during their shopping spree and for each the family member, numbered $1 \leq i \leq F$, list the item numbers $1 \leq N \leq 100$ that they should select.

Sample Input
2
3
72 17
44 23
31 24
1
26
6
64 26
85 22
52 4
99 18
39 13
54 9
4
23
20
20
36

Sample Output:
Test Case 1
Total Price 72
Member Items
     1:   1
Test Case 2
Total Price 568
Member Items
     1:   3 4
     2:   3 6
     3:   3 6
     4:   3 4 6

Note: You will not be collecting experimental running rimes for the Shopping Spree problem.

Submit to TEACH a zipped file containing your code files for problems 3 & 4 and README file. The README file should give instruction on how to compile and run your program.