Programming Exercise 5: Dynamic Programming Instructor: Neil Patrick Del Gallego
Individual

[45 pts] General Instructions: This is a programming exercise for you to test your algorithmic design skills, specifically for implementing **dynamic programming (DP)** solutions. You are to devise a **DP solution** for all problems specified. The most efficient solution for these problems are clearly **DP!** **Not providing a DP solution will have no merit.** For convention and consistency, the mini-program are to be done in JAVA. Please use only 1 java class for this exercise.

1. **[15 pts] Matrix Chain Multiplication**
You are tasked to perform the optimal parenthesizing in a matrix chain multiplication.

Input: The first line contains the number of matrices to multiply (3 ≤ N ≤ 100). Succeeding **N** lines will contain three space-separated digits/characters, **A, B, C**. **A** will be a character that denotes the label of the matrix, followed by its dimension of **B** X **C**.
Output: The output will contain the parentheses of how to perform multiplication of **N** matrices.

Sample Input:	**Sample Output:**
3 A 10 30 B 30 5 C 5 60	(AB)C
4 A 4 10 B 10 3 C 3 12 D 12 20 E 20 7	(AB)((CD)E)

Coding Instructions
For #1, implement the solution in the function seen below. This function will be called from an external class (created by your instructor) to test your code.

```
public static void assignTable(String[] input) {
  //there will be N inputs. Each input is space-separated into A,B and C.
  //Read the input, parse and represent in your own preference.
  //your print statements here
}
```

2. **[15 pts] Party Scheduling**
You introduce a limit for your party budget and try to have the most possible fun with regard to this limit. You inquire beforehand about the entrance fee to each party and estimate how much fun you might have there. The list is readily compiled, but how do you actually pick the parties that give you the most fun and do not exceed your budget?

Write a program which finds this optimal set of parties that offer the most fun. Keep in mind that your budget need not necessarily be reached exactly. Achieve the highest possible fun level, and do not spend more money than is necessary.

Input: The first line of the input specifies your party budget and the number **n** of parties. The following **n** lines contain two numbers each. The first number indicates the **entrance fee** of each party. Parties cost between 5 and 25 PHP. The second number indicates the amount of **fun** of each party, given as an integer number ranging from 0 to 10.

The budget will not exceed 500 PHP and there will be at most 100 parties. All numbers are separated by a single space.

Output: The output must be the sum of the entrance fees and the sum of all fun values of an optimal solution. Both numbers must be separated by a single space.

Sample Input:	**Sample Output:**

50 10
12 3
15 8
16 9
16 6
10 2
21 9
18 4
12 4
17 8
18 9
	49 26

50 10	48 32
13 8	
19 10	
16 8	
12 9	
10 2	
12 8	
13 5	
15 5	
11 7	
16 2	

Coding Instructions
For #2, implement the solution in the function seen below. This function will be called from an external class (created by your instructor) to test your code.

```
public static void partyBudget(String[] inputs) {
  //there will be N inputs. Each input is space-separated into entrance fee, and fun rating.
  //Read the input, parse and represent in your own preference.
  //your print statements here
}
```

3. **[15 pts] Wood Cutter**
You have to cut a wood stick into pieces. A cutting company charges money according to the **length** of the stick being cut. Their procedure of work requires that they only make one cut at a time. It is easy to notice that different selections in the order of cutting can led to different prices. For example, consider a stick of length 10 meters that has to be cut at 2, 4 and 7 meters from one end. There are several choices. One can be cutting first at 2, then at 4, then at 7. This leads to a price of 10 + 8 + 6 = 24 because the first stick was of 10 meters, the resulting of 8 and the last one of 6. Another choice could be cutting at 4, then at 2, then at 7. This would lead to a price of 10 + 4 + 6 = 20, which is a better price. Your boss trusts your computer abilities to find out the minimum cost for cutting a given stick.

Input: The first line of each test case will contain a positive number **l** that represents the length of the stick to be cut. You can assume **$l < 1000$.** The next line will contain the number **n ($n < 50$)** of cuts to be made. The next line consists of **n** positive numbers **c_i ($0 < c_i < l$)** representing the places where the cuts must be done, given in strictly increasing order.

Output: You must print the cost of the optimal solution of the cutting problem, that is the minimum cost of cutting the given stick. Format the output as shown below.

Sample Input:	**Sample Output:**
100	The minimum cutting is 200.
3	
25 50 75	
10	The minimum cutting is 22
4	
4 5 7 8	

Coding Instructions
For #3, implement the solution in the function seen below. This function will be called from an external class (created by your instructor) to test your code.

```
public static void cut (int l, int cuts, int[] places) {
  //your print statements here
}
```

Submission Instructions
- Use only 1 Java class for this exercise. You are free to use as many functions, inner classes or structs to organize your code.
- Follow the coding instructions! Not following the coding instructions will merit deductions.
- Test your code thoroughly and try out different test cases, especially the edge cases (0 or N values). Your instructor will run with at least 3 test cases to check your code.
- Optimize your performance. If your code executes for more than 5 seconds, it will automatically fail the test case! Your code will be tested on an i7 processor with 8GB of RAM.
- Submit this exercise on or before the indicated deadline in Canvas.