CSE 100 Algorithm Design and Analysis Lab 06 (20 points)
Due date: October 25, 23:59

Dynamic Programming: Optimal Matrix Chain Multiplication Order

In this assignment you are asked to implement a dynamic programming algorithm: matrix chain multiplication (chapter 15.2), where the goal is to find the most computationally efficient matrix order when multiplying an arbitrary number of matrices in a row. You can assume that the entire input will be given as integers that can be stored using the standard C++ `int` type and that matrix sizes will be at least 1. You will use "Grade06" to grade your code. Your execution file name must be "MatrixChain.exe". Refer to the previous lab assignments for instructions on how to use the grading tool.

The input has the following format. The first number, $n \geq 1$, in the test case will tell you how many matrices are in the sequence. The first number will be then followed by $n + 1$ numbers indicating the size of the dimensions of the matrices. Recall that the given information is enough to fully specify the dimensions of the matrices to be multiplied.

First, you need to output the minimum number of scalar multiplications needed to multiply the given matrices. Then, print the matrix multiplication sequence, via parentheses, that minimizes the total number of multiplications. Each matrix should be named A#, where # is the matrix number starting at 0 (zero) and ending at $n - 1$. See the examples below.

Examples of input and output

```
Input 1
2
2 3 5
Output 1
30
(A0A1)

Input 2
3
10 100 5 50
Output 2
7500
((A0A1)A2)

Input 3
3
10 30 5 60
Output 3
4500
((A0A1)A2)

Input 4
6
```

```
30 35 15 5 10 20 25
Output 4
15125
((A0(A1A2))((A3A4)A5))
```

Submission Submit the source code (only `MatrixChain.cpp` file) through the assignments page of CatCourses by the deadline. Be careful since CatCourses strictly enforces the assignment deadline. You may be asked to compile, run, and explain the code to the TA to prove that you understand what you wrote.

Grading We provide 10 test cases for you to try your implementation. Each of them is valued at 1 point if executed correctly. We will use 10 additional test cases to check that your implementation is general enough. These are not provided to you and are also valued at 1 point each.

Important Note We will use plagiarism software to detect cheating. Offenders will be subjected to the UCM Academic Honesty Policy which states: *if any violation of the UCM Academic Honesty Policy is suspected in a course, the instructor of record must fill out the Faculty Report for Academic Misconduct.*