Special instructions: You may work either individually or in a group of exactly two persons. Write your solutions out on paper, photocopy them and hand in both the original and the copy to SSO by 12:00 on Friday, 2nd December, 2016. If you are working in a group, one script should be submitted for the group, with both names at the top; both members of the group will then receive the same mark. Clearly write your name(s), student ID number(s) and the words "Comp36111 Sec. B Coursework" on the front (cover) sheet and staple all sheets together. Submitted solutions must be entirely the effort of the persons whose names appear at the top of the submission.

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Advanced Algorithms I: Coursework for Sec. B

Time: This should take you a few hours

Please answer all questions.

Marks will be awarded for clarity and succinctness as well as correctness.

The use of electronic calculators is not recommended.

Section B

We all know how to multiply matrices, don't we? It's a piece of cake. If $A \cdot B = C$, where A has dimensions $\ell \times m$ and B has dimensions $m \times n$, then, writing $a_{i,j}$ for the entry of A in row i and column j, and similarly for B and C, we have

$$c_{i,k} = \sum_{j=1}^{m} a_{i,j} b_{j,k}.$$

So the algorithm

```
mult(A, B, ℓ, m, n)
    for i from 1 to ℓ
        for k from 1 to n
            c_{i,k} ← 0
            for j from 1 to m
                c_{i,k} ← c_{i,k} + a_{i,j} b_{j,k}
    return C
```

works. Clearly this algorithm runs in cubic time. Actually, there are fancier algorithms, but this is the one we shall use.

Remember that matrix multiplication is associative: $(A \cdot B) \cdot C = A \cdot (B \cdot C)$. That is: the bracketing of matrices is irrelevant: we can unambiguously write $A \cdot B \cdot C$. Suppose, however, you have a whole row of matrices to multiply:

$$A_1 \cdot A_2 \cdot \quad \cdots \quad A_{n-1} \cdot A_n.$$

The matrices need not all be square, but their dimensions are all compatible: there exist numbers $\ell_1, \ldots, \ell_{n+1}$ such that each A_i has dimensions, say, $\ell_i \times \ell_{i+1}$ $(1 \leq i < n)$; hence the matrix product makes sense. How are you going to bracket the matrices? Here are some possibilities:

$$A_1 \cdot (A_2 \cdot (A_3 \cdot \quad \cdots \quad (A_{n_1} \cdot A_n)))$$
$$(((A_1 \cdot A_2) \cdot A_3) \quad \cdots \quad A_{n-1}) \cdot A_n$$

and of course many more possibilities besides.

1. Give a *small* example to show that the total number of operations performed depends on how you bracket the matrices. (5 marks)

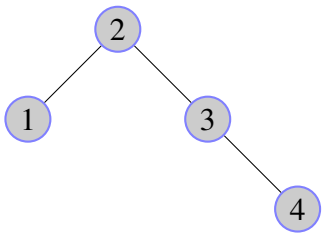
2. Give an algorithm for working out the optimal bracketing of the matrices in a product

$$A_1 \cdot A_2 \cdot \quad \cdots \quad A_{n-1} \cdot A_n,$$

where each A_k has dimensions $\ell_k \times \ell_{k+1}$. The algorithm should output the bracketing in the form of a binary tree having exactly $n-1$ vertices are labelled $1,\ldots,n-1$ in some order. If the root node is i, then the optimal bracketing is

$$(A_1 \cdot \quad \cdots \quad \cdot A_i) \cdot (A_{i+1} \cdot \quad \cdots \quad \cdot A_n).$$

The left subtree of the root gives the optimal top-level bracketing of $(A_1 \cdot \quad \cdots \quad \cdot A_i)$ and the right subtree the optimal top-level bracketing of $(A_{i+1} \cdot \quad \cdots \quad \cdot A_n)$. Thus, for example, if the optimal bracketing of $A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5$ is $((A_1 \cdot A_2) \cdot (A_3 \cdot (A_4 \cdot A_5)))$, then the tree output tree is:



Your algorithm should run in polynomial time in the size of the problem. Hint: use dynamic programming. That is: suppose you worked out the optimal bracketing for some shorter sequences $A_j \ldots, A_k$. How might this help? (10 marks)

3. Give asymptotic complexity bounds for your algorithm. (5 marks)

END OF EXAMINATION