PROGRAM 11: IMPLEMENTATION OF MATRIX CHAIN MULTIPLICATION

Aim: To write a program in C++ for implementing matrix chain multiplication.

Description:

We are given a sequence (chain) <A1, A2, ... ,An> of n matrices to be multiplied, and we wish to compute the product

A1*A2* .... * An

We can evaluate the expression using the standard algorithm for multiplying pairs of matrices as a subroutine once we have parenthesized it to resolve all ambiguities in how the matrices are multiplied together.

Matrix multiplication is associative, and so all parenthesizations yield the same product. A product of matrices is fully parenthesized if it is either a single matrix or the product of two fully parenthesized matrix products, surrounded by parentheses. For example, if the chain of matrices is <A1, A2, A3, A4> then we can fully parenthesize the product

A1*A2*A3*A4 in five distinct ways:

1. (A1(A2 (A3 * A4)))
2. (A1((A2 * A3) A4))
3. ((A1 * A2) (A3 * A4))
4. ((A1 (A2 * A3) A4)
5. (((A1 * A2) A3) A4)

How we parenthesize a chain of matrices can have a dramatic impact on the cost of evaluating the product. Consider first the cost of multiplying two matrices.

So here is the Formula we will be used for solving our problem in an optimized way,

$$B(i,j) = \begin{cases} 0 \text{ if } i = j \\ \min\limits_{k=i}^{j-1} \left\{ B(i,k) + B(k+1,j) + r_i \cdot c_k \cdot c_j \right. \end{cases}$$

Algorithm:

matOrder(array, n)

Input: List of matrices, the number of matrices in the list.

Output: Minimum number of matrix multiplication.

Step 1: First, it will divide the matrix sequence into two subsequences.

Step 2: You will find the minimum cost of multiplying out each subsequence.

Step 3: You will add these costs together and in the price of multiplying the two result matrices.

Step 4: These procedures will be repeated for every possible matrix split and calculate the minimum.

Program:

```
#include<iostream>
```

```
#include <climits>
using namespace std;
int matOrder(int array[], int n){
int minMul[n][n]; //holds the number of scalar multiplication needed
for (int i=1; i<n; i++)
minMul[i][i] = 0; //for multiplication with 1 matrix, cost is 0
for (int length=2; length<n; length++){ //find the chain length starting from 2
for (int i=1; i<n-length+1; i++){
int j = i+length-1;
minMul[i][j] = INT_MAX; //set to infinity
for (int k=i; k<=j-1; k++){
//store cost per multiplications
int q = minMul[i][k] + minMul[k+1][j] + array[i-1]*array[k]*array[j];
if (q < minMul[i][j])
        minMul[i][j] = q;
}
}
}
        return minMul[1][n-1];
}
int main()
{
int arr[] = {1, 2, 3, 4};
int size = 4;
```

```
cout << "Minimum number of matrix multiplications:
"<<matOrder(arr,size);
}
```

Output:

Minimum number of matrix multiplications: 18

Result:

Thus a C++ program has been written and executed for implementing MaxHeap.