EXPERIMENT 2.1

STUDENT NAME: Yash Kumar **UID: 20BCS9256**

BRANCH: CSE **SECTION/GROUP: 616-B**

SUBJECT NAME: DAA LAB **SUBJECT CODE: 20CSP-312**

AIM/OVERVIEW OF THE PRACTICAL:

Code and analyze to find an optimal solution to matrix chain multiplication using dynamic programming.

TASK TO BE DONE/WHICH LOGISTICS USED:

Find an optimal solution to matrix chain multiplication using dynamic programming.

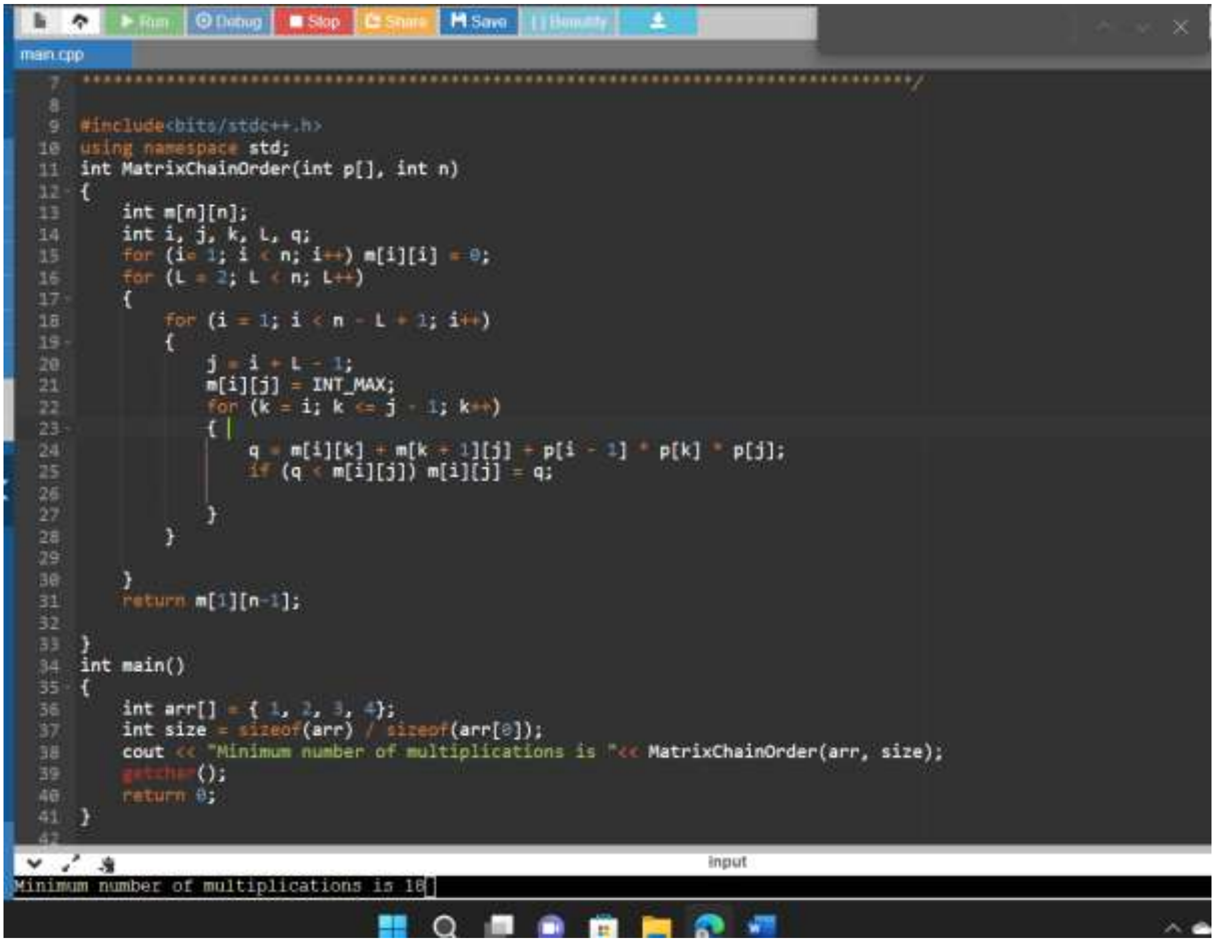
ALGORITHM / FLOWCHART:

1) Start
2) Iterate from l = 2 to N-1 which denotes the length of the range:
3) Iterate from i = 0 to N-1:
4) Find the right end of the range (j) having l matrices.
5) Iterate from k = i+1 to j which denotes the point of partition.
6) Multiply the matrices in range (i, k) and (k, j).
7) This will create two matrices with dimensions arr[i-1]*arr[k] and arr[k]*arr[j].
8) The number of multiplications to be performed to multiply these two matrices (say X) are arr[i-1]*arr[k]*arr[j].
9) The total number of multiplications is dp[i][k]+ dp[k+1][j] + X. 10) The value stored at dp[1][N-1] is the required answer.

10) End.

STEPS FOR EXPERIMENT / PRACTICAL / CODE:

```
#include<bits/stdc++.h>
using namespace std;
int MatrixChainOrder(int p[], int n)
{
    int m[n][n];
    int i, j, k, L, q;
    for (i= 1; i < n; i++) m[i][i] = 0;
    for (L = 2; L < n; L++)
    {
        for (i = 1; i < n - L + 1; i++)
        {
            j = i + L - 1;
            m[i][j] = INT_MAX;
            for (k = i; k <= j - 1; k++)
            {
                q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];
                if (q < m[i][j]) m[i][j] = q;

            }
        }

    }
    return m[1][n-1];

}
int main()
{
    int arr[] = { 1, 2, 3, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    cout << "Minimum number of multiplications is "<< MatrixChainOrder(arr, size);
    getchar();
    return 0;
}
```



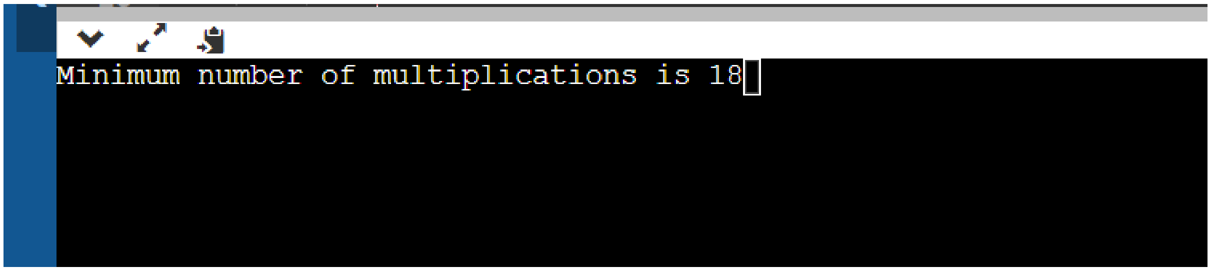
```
**************************************************************************************/


#include<bits/stdc++.h>
using namespace std;
int MatrixChainOrder(int p[], int n)
{
    int m[n][n];
    int i, j, k, L, q;
    for (i= 1; i < n; i++) m[i][i] = 0;
    for (L = 2; L < n; L++)
    {
        for (i = 1; i < n - L + 1; i++)
        {
            j = i + L - 1;
            m[i][j] = INT_MAX;
            for (k = i; k <= j - 1; k++)
            {
                q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];
                if (q < m[i][j]) m[i][j] = q;

            }
        }

    }
    return m[1][n-1];

}
int main()
{
    int arr[] = { 1, 2, 3, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    cout << "Minimum number of multiplications is "<< MatrixChainOrder(arr, size);
    getchar();
    return 0;
}
```

```
Minimum number of multiplications is 18
```

OBSERVATIONS/DISCUSSIONS/ COMPLEXITY ANALYSIS:

Time Complexity: O (N^3)
Auxiliary Space: O(N^2)

OUTPUT:



```
Minimum number of multiplications is 18
```