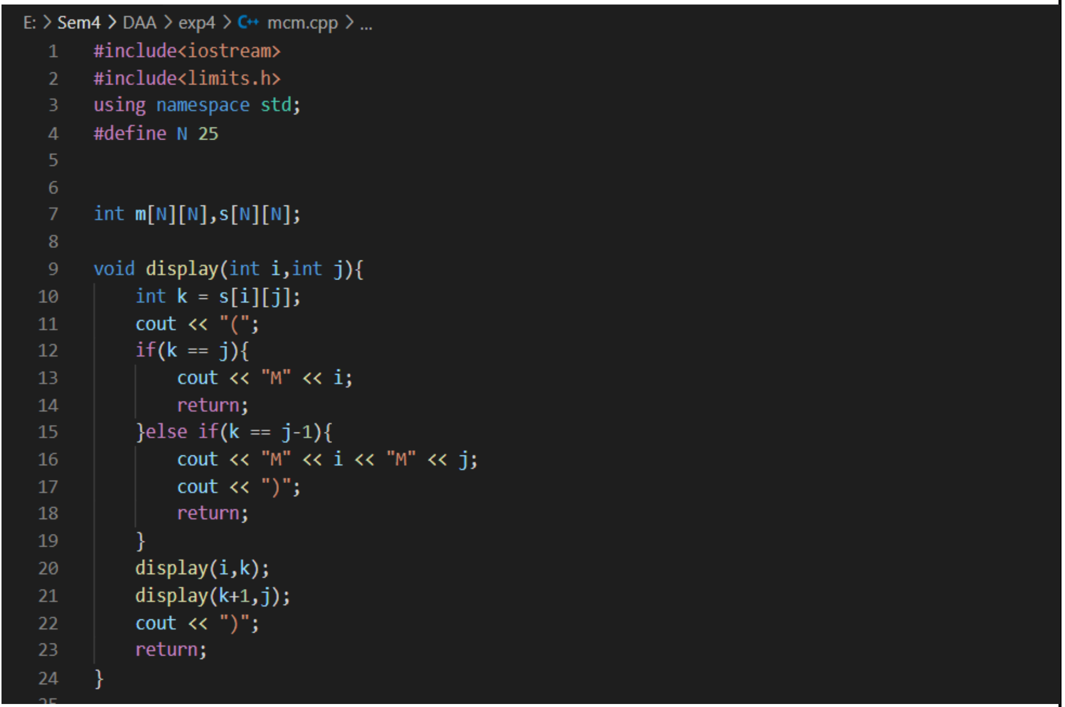
NAME:	Bhavesh Prashant Chaudhari
UID:	2021300018
SUBJECT	DAA
EXPERIMENT NO :	04
AIM:	To use DP to find minimum scalar multiplication required for a chain of matrices.
PROBLEM STATEMENT 1:	
ALGORITHM	Two matrices of size m*n and n*p when multiplied, they generate a matrix of size m*p and the number of multiplications performed are m*n*p. Now, for a given chain of N matrices, the first partition can be done in N-1 ways. For example, sequence of matrices A, B, C and D can be grouped as (A)(BCD), (AB)(CD) or (ABC)(D) in these 3 ways. So a range [i, j] can be broken into two groups like {[i, i+1], [i+1, j]}, {[i, i+2], [i+2, j]}, . . . , {[i, j-1], [j-1, j]}. Each of the groups can be further partitioned into smaller groups and we can find the total required multiplications by solving for each of the groups. The minimum number of multiplications among all the first partitions is the required answer.

Psuedo Code-

```
MATRIX-CHAIN-ORDER(p,n)
let m[1:n,1:n] and s[1:n-1,2:n] be new tables
for i=1 to n
  m[i,j] = 0
  for l=2 to n
    for i=1 to n-l+1
      j=i+l-1
      m[i,j] = INFINITE
      for k=i to j-1
        q=m[i,k]+m[k+1,j]+p[i-1]*p[k]*p[j]
        if q<m[i,j]
          m[i,j]=q
          s[i,j]=k
return m and s
```

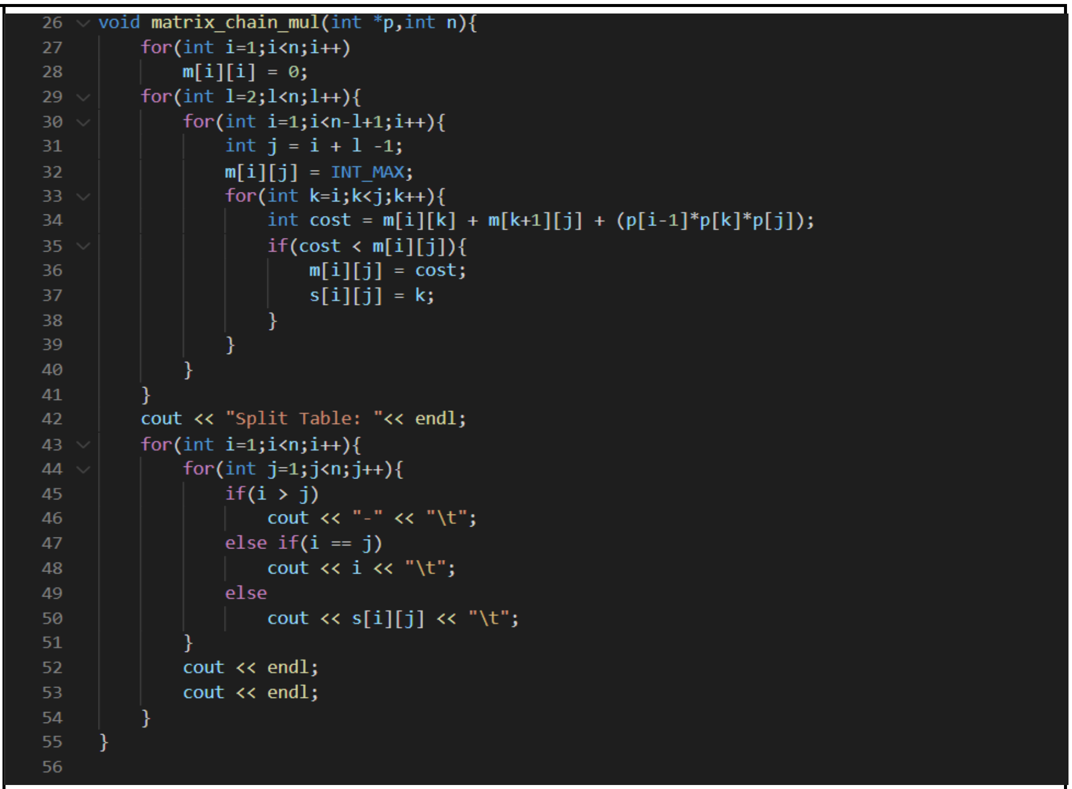
PROGRAM:



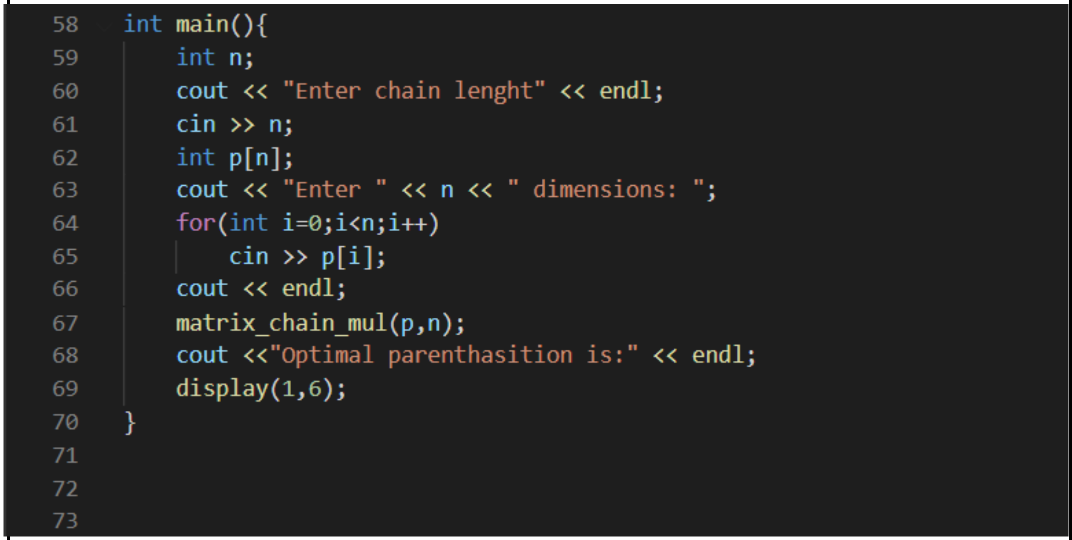
```cpp
#include<iostream>
#include<limits.h>
using namespace std;
#define N 25


int m[N][N],s[N][N];

void display(int i,int j){
    int k = s[i][j];
    cout << "(";
    if(k == j){
        cout << "M" << i;
        return;
    }else if(k == j-1){
        cout << "M" << i << "M" << j;
        cout << ")";
        return;
    }
    display(i,k);
    display(k+1,j);
    cout << ")";
    return;
}
```

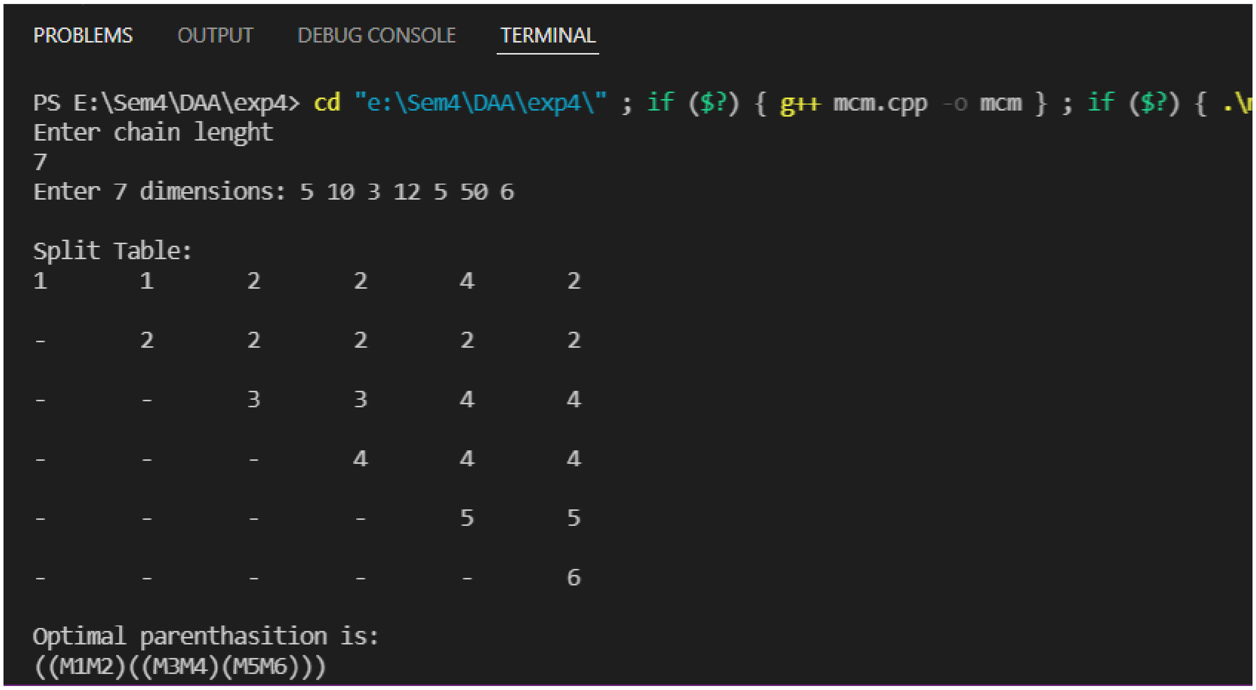


```
void matrix_chain_mul(int *p,int n){
    for(int i=1;i<n;i++)
        m[i][i] = 0;
    for(int l=2;l<n;l++){
        for(int i=1;i<n-l+1;i++){
            int j = i + l -1;
            m[i][j] = INT_MAX;
            for(int k=i;k<j;k++){
                int cost = m[i][k] + m[k+1][j] + (p[i-1]*p[k]*p[j]);
                if(cost < m[i][j]){
                    m[i][j] = cost;
                    s[i][j] = k;
                }
            }
        }
    }
    cout << "Split Table: "<< endl;
    for(int i=1;i<n;i++){
        for(int j=1;j<n;j++){
            if(i > j)
                cout << "-" << "\t";
            else if(i == j)
                cout << i << "\t";
            else
                cout << s[i][j] << "\t";
        }
        cout << endl;
        cout << endl;
    }
}
```



```
int main(){
    int n;
    cout << "Enter chain lenght" << endl;
    cin >> n;
    int p[n];
    cout << "Enter " << n << " dimensions: ";
    for(int i=0;i<n;i++)
        cin >> p[i];
    cout << endl;
    matrix_chain_mul(p,n);
    cout <<"Optimal parenthasition is:" << endl;
    display(1,6);
}
```

RESULT (SNAPSHOT)



```
PS E:\Sem4\DAA\exp4> cd "e:\Sem4\DAA\exp4\" ; if ($?) { g++ mcm.cpp -o mcm } ; if ($?) { .\
Enter chain lenght
7
Enter 7 dimensions: 5 10 3 12 5 50 6

Split Table:
1       1       2       2       4       2

-       2       2       2       2       2

-       -       3       3       4       4

-       -       -       4       4       4

-       -       -       -       5       5

-       -       -       -       -       6

Optimal parenthasition is:
((M1M2)((M3M4)(M5M6)))
```

CONCLUSION:

Through this experiment I learned how to used DP approach to find minimum scalar multiplication for a chain of matrix.

Eg Consider a chain of matrix <10,100,5,50>

M1 = 10x100, M2=100x5, M3=5x50

M1(M2.M3) = (10*100*50) + (100*5*50) = 75000

(M1.M2)M3 = (10*100*5) + (10*5*50) = 7500

We can see that second one gives minimum scalar multiplication