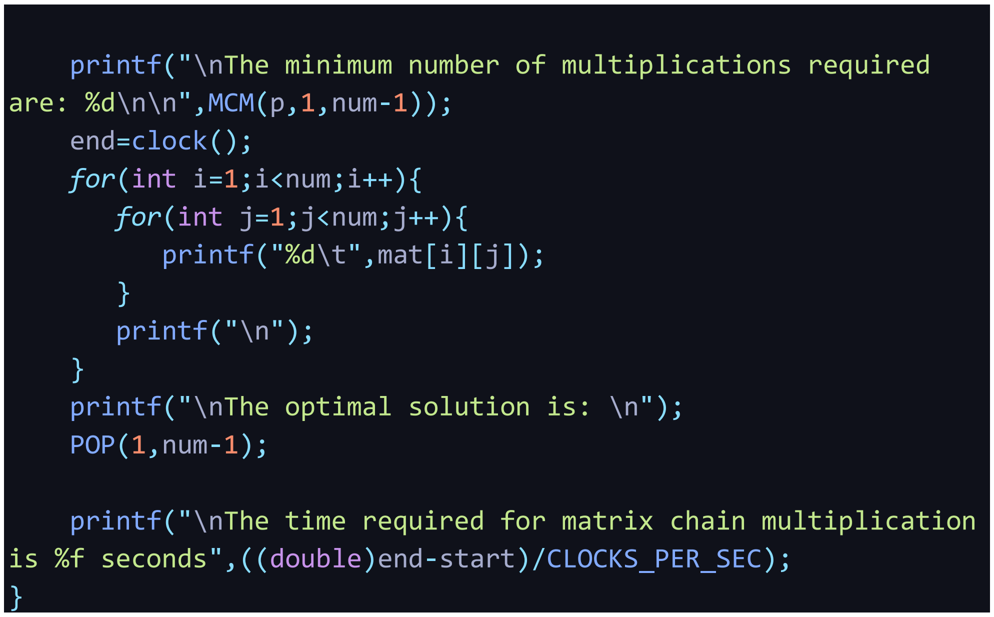
Name	MAAZ SHAIKH
UID no.	2021700059
Experiment No.	05

AIM:	To implement dynamic algorithms
PROBLEM STATEMENT :	The aim of this experiment is two-fold. First, it finds the efficient way of multiplying a sequence of k matrices (called Matrix Chain Multiplication) using Dynamic Programming. The chain of multiplication M1 x M2 x M3 x M4 x...x Mk may be computed in $(2N)!/((N + 1)!\ N!)$ = $(2N\ N)\ /(N + 1)$ ways due to associative property where $NN = kk - 1$ of matrix multiplication
ALGORITHM/ THEORY:	Given a sequence of matrices, find the most efficient way to multiply these matrices together. The problem is not actually to perform the multiplications, but merely to decide in which order to perform the multiplications. We have many options to multiply a chain of matrices because matrix multiplication is associative. In other words, no matter how we parenthesize the product, the result will be the same. For example, if we had four matrices A, B, C, and D, we would have: (ABC)D = (AB)(CD) = A(BCD) = ... However, the order in which we parenthesize the product affects the number of simple arithmetic operations needed to compute the product, or the efficiency. For example, suppose A is a 10 × 30 matrix, B is a 30 × 5 matrix, and C is a 5 × 60 matrix. Then, (AB)C = (10×30×5) + (10×5×60) = 1500 + 3000 = 4500 operations A(BC) = (30×5×60) + (10×30×60) = 9000 + 18000 = 27000 operations. Clearly the first parenthesization requires less number of operations.

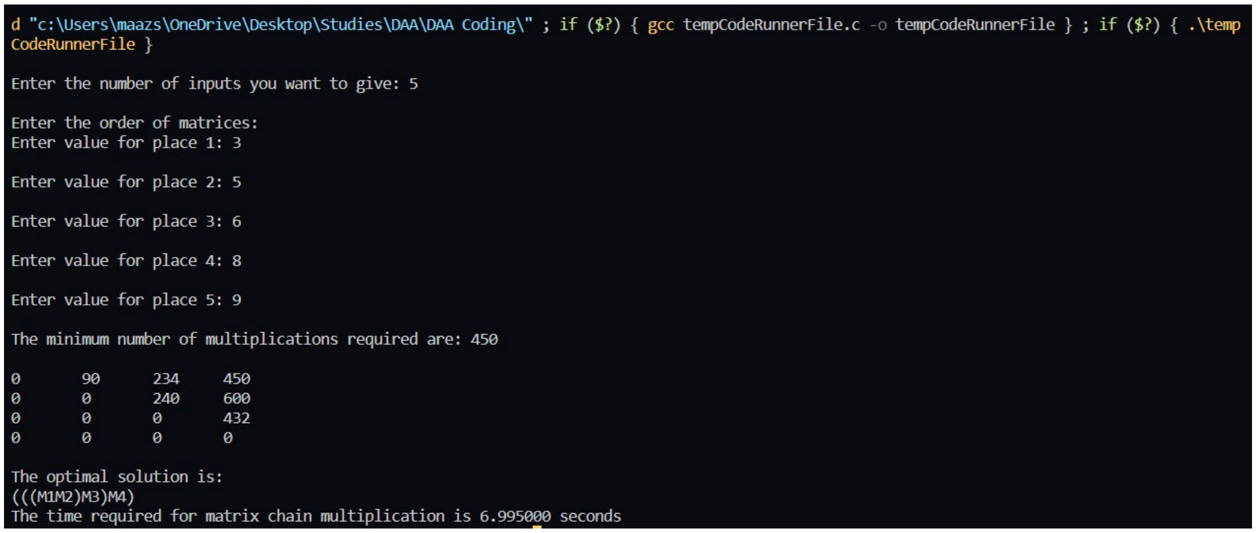
PROGRAM:

```c
#include<stdio.h>
#include<time.h>
#include<stdlib.h>

int mat[100][100],s[100][100],count=0;
int MCM(int p[], int i, int j){
    if(i==j){
      mat[i][j] = 0;
      return 0;
    }
    mat[i][j] = 30000;
    for(int k=i; k<j; k++){
       count = MCM(p,i,k) + MCM(p,k+1,j) + p[i-1]*p[k]*p[j];
       if(count<mat[i][j]){
         mat[i][j] = count;
         s[i][j] = k;
       }
    }
    return mat[i][j];
}
void POP(int i,int j){
    if(i==j)
      printf("M%d",i);
    else{
      printf("(");
      POP(i,s[i][j]);
      POP(s[i][j]+1,j);
      printf(")");
    }
}
void main(){
    int num;
    clock_t start,end;


    printf("\nEnter the number of inputs you want to give: ");
    scanf("%d",&num);
    int p[num];
    printf("\nEnter the order of matrices: ");
    start=clock();
    for(int i=0;i<num;i++){
       printf("\nEnter value for place %d: ",i+1);
       scanf("%d",&p[i]);
    }
```



```
    printf("\nThe minimum number of multiplications required
are: %d\n\n",MCM(p,1,num-1));
    end=clock();
    for(int i=1;i<num;i++){
       for(int j=1;j<num;j++){
          printf("%d\t",mat[i][j]);
       }
       printf("\n");
    }
    printf("\nThe optimal solution is: \n");
    POP(1,num-1);

    printf("\nThe time required for matrix chain multiplication
is %f seconds",((double)end-start)/CLOCKS_PER_SEC);
}
```

RESULT:



```
d "c:\Users\maazs\OneDrive\Desktop\Studies\DAA\DAA Coding\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\temp
CodeRunnerFile }

Enter the number of inputs you want to give: 5

Enter the order of matrices:
Enter value for place 1: 3

Enter value for place 2: 5

Enter value for place 3: 6

Enter value for place 4: 8

Enter value for place 5: 9

The minimum number of multiplications required are: 450

0       90      234     450
0       0       240     600
0       0       0       432
0       0       0       0

The optimal solution is:
(((M1M2)M3)M4)
The time required for matrix chain multiplication is 6.995000 seconds
```

CONCLUSION: I learned matrix chain multiplication and how it is used to find the minimum cost required to multiply given sequence of matrices.