Name	Siddiqui Hawaiza Sabeel
UID no.	2022701010
Experiment No.	5
SUBJECT	DAA

AIM:	**Dynamic Programming - Matrix Chain Multiplication**
Program 1	
PROBLEM STATEMENT:	Use the Dynamic Programming method to find the optimal way to multiply(parenthesize) the matrices to find the minimum number of multiplications required to solve the matrix.
ALGORITHM/ THEORY:	**Matrix chain multiplication** (or the **matrix chain ordering problem**) is an optimization problem concerning the most efficient way to multiply a given sequence of matrices. The problem is not actually to *perform* the multiplications, but merely to decide the sequence of the matrix multiplications involved. The problem may be solved using dynamic programming. There are many options because matrix multiplication is associative. In other words, no matter how the product is parenthesized, the result obtained will remain the same. For example, for four matrices A, B, C, and D, there are five possible options: $((AB)C)D = (A(BC))D = (AB)(CD) = A((BC)D) = A(B(CD))$. Although it does not affect the product, the order in which the terms are parenthesized affects the number of simple arithmetic operations needed to compute the product, that is, the computational complexity. The straightforward multiplication of a matrix that is $X \times Y$ by a matrix that is $Y \times Z$ requires XYZ ordinary multiplications and $X(Y - 1)Z$ ordinary additions. In this context, it is typical to use the number of ordinary multiplications as a measure of the runtime complexity. If A is a 10×30 matrix, B is a 30×5 matrix, and C is a 5×60 matrix, then computing $(AB)C$ needs $(10\times30\times5) + (10\times5\times60) = 1500 + 3000 = 4500$ operations, while computing $A(BC)$ needs $(30\times5\times60) + (10\times30\times60) = 9000 + 18000 = 27000$ operations. Clearly the first method is more efficient. With this information, the problem statement can be refined as "how to determine the optimal parenthesization of a

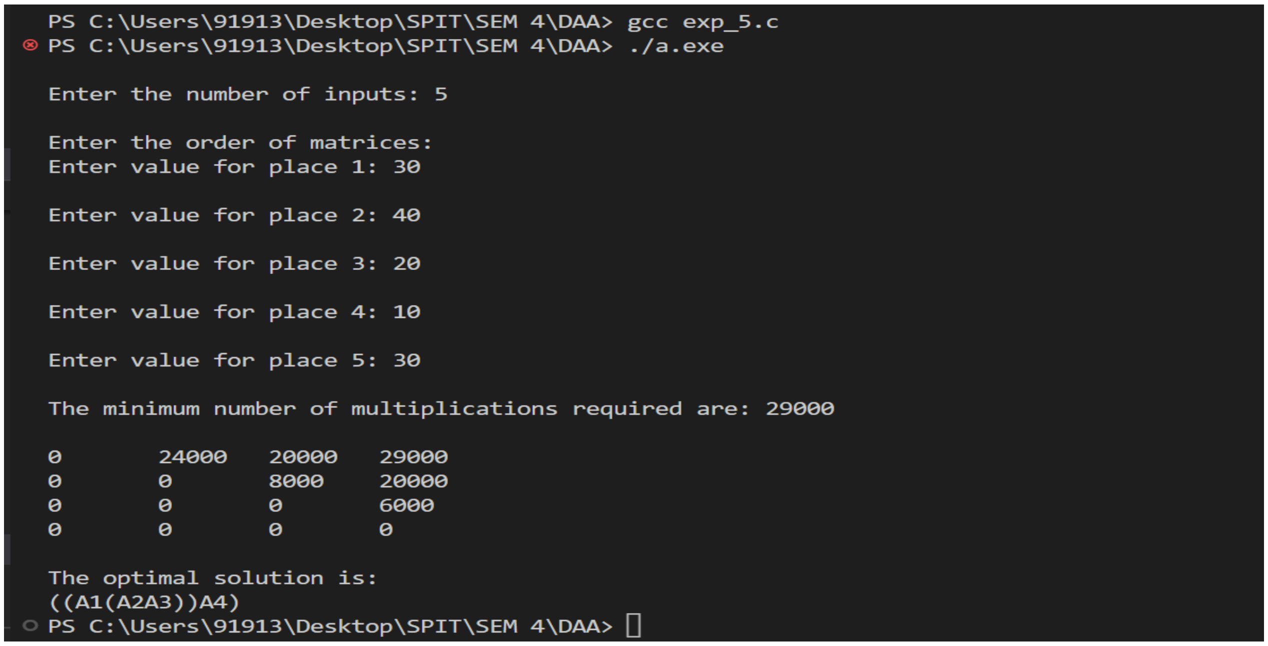
product of *n* matrices?" Checking each possible parenthesization (brute force) would require a run-time that is exponential in the number of matrices, which is very slow and impractical for large *n*. A quicker solution to this problem can be achieved by breaking up the problem into a set of related subproblems.

1. Take the sequence of matrices and separate it into two subsequences.
2. Find the minimum cost of multiplying out each subsequence.
3. Add these costs together, and add in the cost of multiplying the two result matrices.
4. Do this for each possible position at which the sequence of matrices can be split, and take the minimum over all of them.

PROGRAM:

```c
#include<stdio.h>

int mat[100][100];
int 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("A%d",i);
    else{
      printf("(");
      POP(i,s[i][j]);
      POP(s[i][j]+1,j);
      printf(")");
    }
}

void main(){
    int num;
    printf("\nEnter the number of inputs: ");
    scanf("%d",&num);
    int p[num];
    printf("\nEnter the order of matrices: ");
    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));
    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);
}
```

RESULT:



```
  PS C:\Users\91913\Desktop\SPIT\SEM 4\DAA> gcc exp_5.c
⊗ PS C:\Users\91913\Desktop\SPIT\SEM 4\DAA> ./a.exe

  Enter the number of inputs: 5

  Enter the order of matrices:
  Enter value for place 1: 30

  Enter value for place 2: 40

  Enter value for place 3: 20

  Enter value for place 4: 10

  Enter value for place 5: 30

  The minimum number of multiplications required are: 29000

  0       24000   20000   29000
  0       0       8000    20000
  0       0       0       6000
  0       0       0       0

  The optimal solution is:
  ((A1(A2A3))A4)
○ PS C:\Users\91913\Desktop\SPIT\SEM 4\DAA> 
```

CONCLUSION:	I understood the dynamic programming approach to solve the Matrix Chain multiplication problem