Experiment 7

Aim:Perform Matrix Chain Multiplication Problem (MCM) using Dynamic Programming.

7.1 CO Attained: CO2 and CO4

7.2Objective:

Chain matrix multiplication is an optimization problem that can be solvedusing dynamic programming. Given a sequence of matrices, the objective is tofind out the most efficient way to multiply the matrices. The efficiencyparameter is number of multiplications steps.

7.3Resources: Turbo c/Dev C++

7.4Program Logic:

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.

```
MATRIX-CHAIN-ORDER(p)
 1  n = p.length − 1
 2  let m[1..n, 1..n] and s[1..n − 1, 2..n] be new tables
 3  for i = 1 to n
 4      m[i, i] = 0
 5  for l = 2 to n                    // l is the chain length
 6      for i = 1 to n − l + 1
 7          j = i + l − 1
 8          m[i, j] = ∞
 9          for k = i to j − 1
10              q = m[i, k] + m[k + 1, j] + p_{i−1} p_k p_j
11              if q < m[i, j]
12                  m[i, j] = q
13                  s[i, j] = k
14  return m and s
```

7.5 Procedure:

1. Create Open Dev C++/C and write a program after that save the program with the .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

7.6 Program Code:

// This code was implemented using Algorithm in the Coremen book

```
#include<stdio.h>
#include<limits.h>
// Matrix Ai has dimension p[i-1] x p[i] for i = 1..n
int MatrixChainMultiplication(int p[], int n)
{
   int m[n][n];
   int i, j, k, L, q;

   for (i=1; i<n; i++)
      m[i][i] = 0;   //number of multiplications are 0(zero) when there is only one matrix
```

```
   //Here L is chain length. It varies from length 2 to length n.
   for (L=2; L<n; L++)
   {
      for (i=1; i<n-L+1; i++)
      {
         j = i+L-1;
         m[i][j] = INT_MAX;  //assigning to maximum value

         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;  //if number of multiplications found less that number will
beupdated.
            }
         }
      }
   }
   return m[1][n-1];   //returning the final answer which is M[1][n]
}
int main()
{
   int n,i;
   printf("Enter number of matrices\n");
   scanf("%d",&n);
   n++;
   int arr[n];
   printf("Enter dimensions \n");
   for(i=0;i<n;i++)
   {
      printf("Enter d%d :: ",i);
      scanf("%d",&arr[i]);
   }
   int size = sizeof(arr)/sizeof(arr[0]);

   printf("Minimum number of multiplications is %d ", MatrixChainMultiplication(arr,
size));

   return 0;
}
```

7.7Conclusion:

```
Enter number of matrices
3
Enter dimensions
Enter d0 :: 4
Enter d1 :: 3
Enter d2 :: 2
Enter d3 :: 3
Minimum number of multiplications is 48
```

7.8Analysis:

A simple inspection of the nested loop structure of MATRIX-CHAIN-ORDER yields a running time of O(n3) for the algorithm.

7.9Lab Viva Questions:

1. What is the most efficient approach to solve this problem?
2. Define Matrix Chain Multiplication.
3. Define the rule of Matrix Chain Multiplication.