Name	Aditi Nilesh Bhutada
UID no.	2021700009
Experiment No.	5

AIM:	Implement matrix chain multiplication using dynamic programming
PROGRAM	
THEORY:	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.
INPUT:	Input nuumber of matrices and generate random dimensions

ALGORITHM:

```
MatrixChainMultiplication(mat[], low , high):
   // 0 steps are required if low equals high
   // case of only 1 sized sub-problem.
   If(low=high):
      return 0

   // Initialize minCost to a very big number.
   minCost = Infinity

   // Iterate from k = low to k = high-1
   For(k=low to high-1):
      /*
       Cost = Cost of Multiplying chain on left side +
             Cost of Multiplying chain on right side +
             Cost of Multiplying matrix obtained from left
             and right side.
      */
      cost=MatrixChainMultiplication(mat, low, k)+
         MatrixChainMultiplication(mat, low+1, high)+
         mat[low-1]*mat[k]*mat[high]

      // Update the minCost if cost<minCost.
      If(cost<minCost):
         minCost=cost

   return minCost
```

PROGRAM:

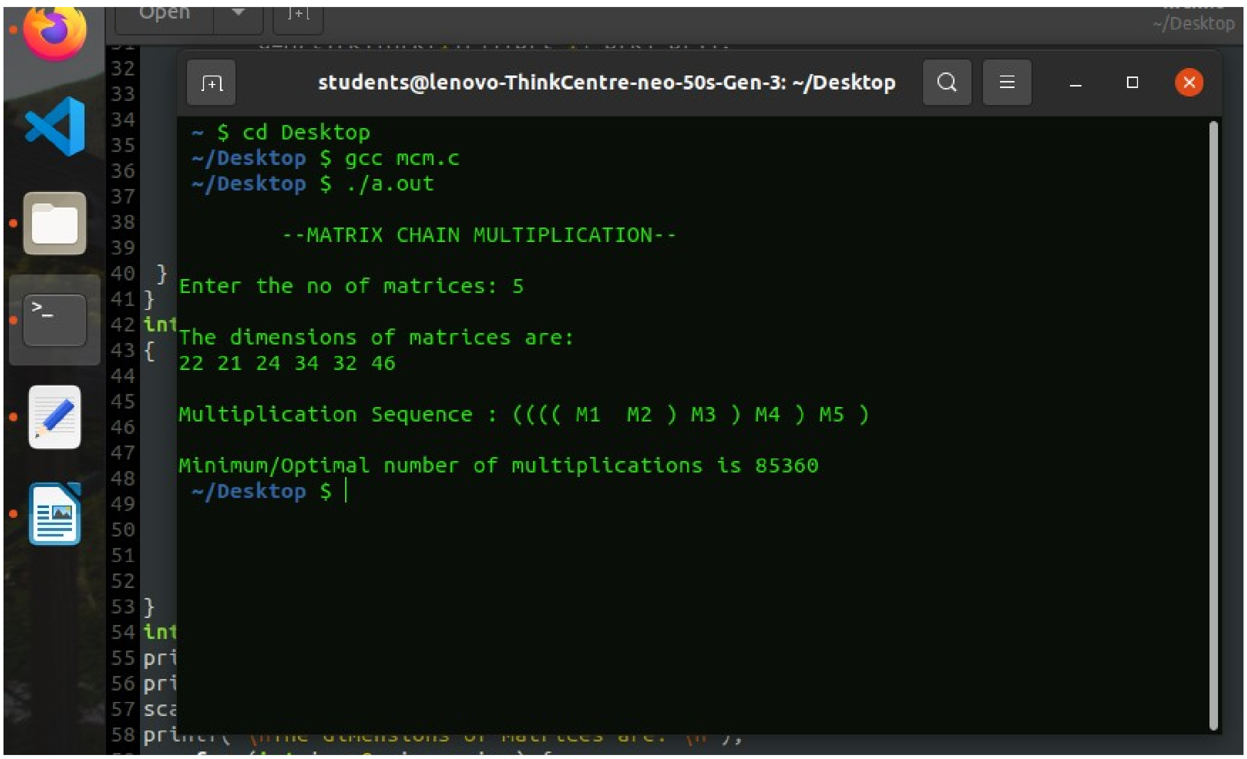
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <limits.h>


int s[20][20],m[20][20],p[20];
int n;
void print(int i,int j){
if (i == j)
printf(" M%d ",i);
else
  {
    printf("(");
    print(i, s[i][j]);
    print(s[i][j] + 1, j);
    printf(")");
```

	`` } } void multiply(){ int q,k; for(int i=n;i>0;i--) { for(int j=i;j<=n;j++) { if(i==j) m[i][j]=0; else { for(int k=i;k<j;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; s[i][j]=k; } } } } } } int chain(int p[], int i, int j) { if(i == j) return 0; int k,min=INT_MAX,count=0; for (k = i; k < j; k++) { count = chain(p, i, k) + chain(p, k + 1, j) + p[i - 1] * p[k] * p[j]; if (count < min) min = count; } return min; } int main(){ printf("\n\t--MATRIX CHAIN MULTIPLICATION--\n") ```

```
printf("\nEnter the no of matrices: ");
scanf("%d",&n);
printf("The dimensions of matrices are: \n");
    for (int i = 0; i<=n; i++) {
        p[i]= (rand()%(46 - 15 + 1)) + 15;
        printf("%d ", p[i]);
    }
for(int i=1;i<=n;i++)
for(int j=i+1;j<=n;j++)
{
 m[i][i]=0;
 m[i][j]=INT_MAX;
 s[i][j]=0;
}
multiply();
printf("\nMultiplication Sequence : ");
print(1,n);
printf("\nMinimum/Optimal number of multiplications is %d\n",chain(p, 1, n));
return 0;
}
```


OUTPUT:



RESULT ANALYSIS:	**Time Complexity -** We are using three nested for loops, each of which is iterating roughly O(n) times. Hence, the overall time complexity is O(n3). **Space Complexity -** We are using an auxiliary dp array of dimensions, (n−1)×(n−1) hence space complexity is O(n2) In the matrix chain multiplication problem, the minimum number of multiplication steps required to multiply a chain of matrices has been calculated. • Determining the minimum number of steps required can highly speed up the

	multiplication process. • It takes O(n3) time and O(n2) auxiliary space to calculate the minimum number of steps required to multiply a chain of matrices using the dynamic programming method. • It is very important to decide the order of multiplication of matrices to perform the task efficiently. • So the minimum number of multiplication steps required to multiply a chain matrix of length n has been computed. • Trying out all the possibilities recursively is very time consuming hence the method of dynamic programming is used to find the same. • The minimum number of steps required to multiply a chain of matrices can be found in O(n3) time complexity using dynamic programming.
CONCLUSION:	In this experiment I understood about how to find the minimalcost of matrix chain multiplication using dynamic programming