Name	DARSHIT BHAGTANI
UID	2021700006
Experiment No.	5

AIM:	Matrix Chain Multiplication
PROBLEM STATEMENT :	Apply the concept of dynamic programming to solve the problem of finding the minimum cost i.e. multiplications required to perform Matrix Chain Multiplications
ALGORITHM/ THEORY:	Matrix Chain Multiplication can be solved using dynamic programming. We can define the minimum number of scalar multiplications needed to iteratively compute the product of a chain of matrices. We start with sub chains of length 1 and then compute the minimum cost for sub chains of increasing length until we have the minimum cost for the entire chain. The time complexity of this algorithm is O(n^3), where n is the number of matrices in the chain. **Algorithm**: 1. Define the subproblem: Find the minimum number of scalar multiplications needed to compute the product of a chain of matrices. 2. Find the recurrence relation: Let M[i,j] be the minimum number of scalar multiplications needed to compute the product of the chain of matrices from matrix i to matrix j. We can define M[i,j] recursively as follows: M[i,j] = min(M[i,k] + M[k+1,j] + a[i-1] x a[k] x a[j]) for $i \leq k < j$ 3. Initialize the base case: M[i,i] = 0 for $1 \leq i \leq n$, where n is the number of matrices in the chain. 4. Solve the subproblems: Compute the minimum cost for subchains of increasing length until we have the minimum cost for the entire chain. 5. Return the final answer: The minimum cost for the entire chain is 1. stored in M[1,n], where n is the number of matrices in the chain.

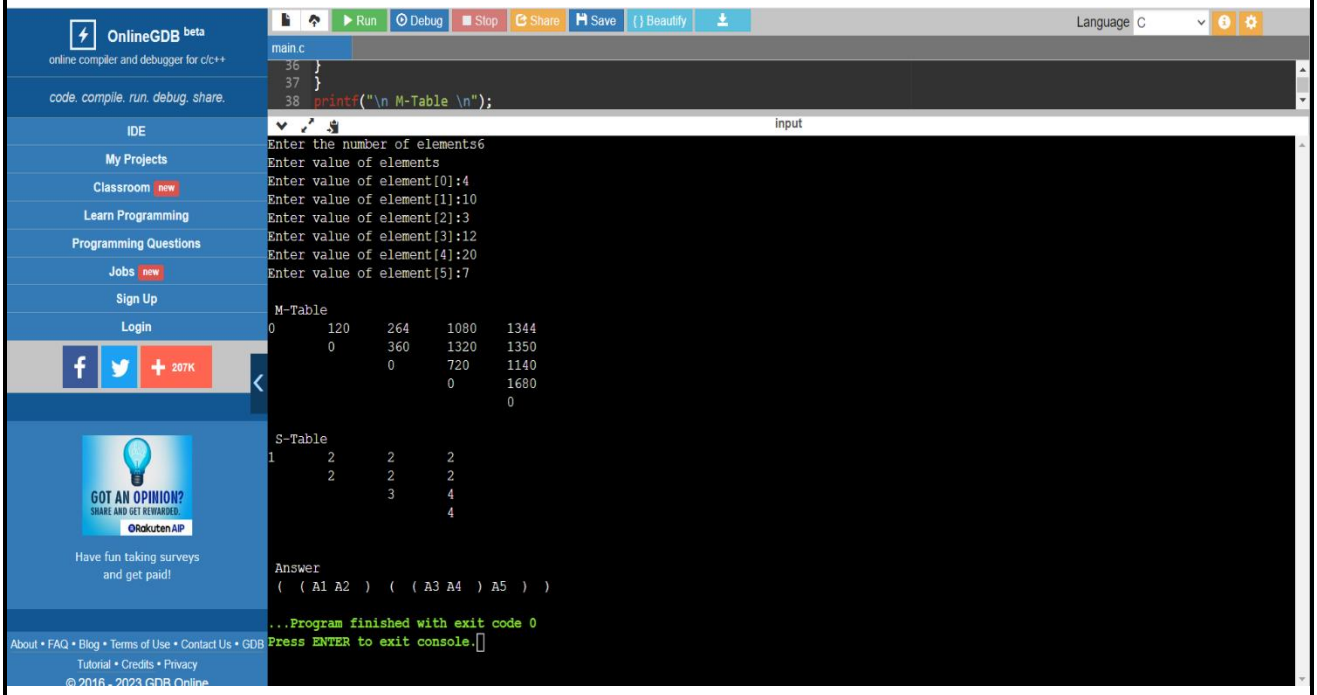
PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void optimal(int s[10][10],int i,int j);
void mcm(int n, int p[10])
{
int k,i,j,m[10][10],temp,s[10][10];
for(j=1;j<=n;j++)
{
for(i=n;i>=1;i--)
{
if(i==j)
{
m[i][j]=0;
}
else if(i<j)
{
m[i][j]=9999;
for(k=i;k<=j-1;k++)
{
temp=m[i][k]+m[k+1][j]+(p[i-1]*p[j]*p[k]);
if(temp<m[i][j])
{
m[i][j]=temp;
s[i][j]=k;
}
}
}
}
}
printf("\n M-Table \n");
for(i=1;i<n;i++)
{
for(j=1;j<n;j++)
{
if(i>j)
{
printf("\t");
}
else
{
printf("%d\t",m[i][j]);
}
}
printf("\n");
}
printf("\n S-Table \n");
for(i=1;i<n;i++)
{
for(j=2;j<n;j++)
{
if(i>=j)
{
printf("\t");
}
```

```
 else
{
printf("%d\t",s[i][j]);
}
}
printf("\n");
}
printf("\n Answer \n");
optimal(s,1,n-1);
}
void optimal(int s[10][10],int i,int j)
{
if(i==j)
{
printf("A%d ",i);
}
else
{
printf(" ( ");
optimal(s,i,s[i][j]);
optimal(s,s[i][j]+1,j);
printf(" ) ");
}
}
void main()
{
int n,p[10],i;

printf("Enter the number of elements");
scanf("%d",&n);
printf("Enter value of elements\n");
for(i=0;i<n;i++)
{
printf("Enter value of element[%d]:",i);
scanf("%d",&p[i]);
}
mcm(n,p);
getch();
}
```

RESULT:



CONCLUSION:	Successfully carried out matrix chain multiplication and displayed the parenthesizing as well