|  |  |
| --- | --- |
| **Experiment No.** | **4** |
| **Aim** | **Experiment using dynamic programming approach (Matrix Chain Multiplication)** |
| **Name** | **Rucha Sudhir Kulkarni** |
| **UID No.** | **2021300067** |
| **Class & Division** | **SE Computer Engineering (Div:A)(Batch:D)** |
| **Date of Performance:** | **29.03.2023** |
| **Date of Submission:** | **04.04.2023** |

**Aim: – To implement dynamic algorithms to optimally parenthesize a matrix chain.**

**Algorithm:**

1. Start
2. Input number of dimensions(10)
3. Print the randomly generated dimensions.
4. Create 2D matrices m and s.
5. Put all m[i][i]=0=s[i][j] and m[i][j]=infinity
6. Call the function matmultiply() and print\_optimal(i,j) to print the optimal parenthesization.
7. Stop
8. Function matmultiply():
   1. If i==j put m[i][j]=0
   2. Else
   3. Compute q=m[i][k]+m[k+1][j]+p[i-1]p[k]p[j] for all i<=k<j
   4. If q<m[i][j], store the k value in s[i][j] and q value in m[i][j]
9. Function print\_optimal(int i,int j):
   1. If i==j, print “ A I”
   2. Else
   3. Print opening bracket
   4. Recursively call print\_optimal(i,s[i][j]) and print\_optimal(s[i][j]+1,j)
   5. Print closing bracket

**Program:**

#include <stdio.h>

#include<stdlib.h>

long int m[20][20];

int s[20][20];

int p[20],i,j,n;

void print\_optimal(int i,int j)

{

if (i == j)

printf(" A%d ",i);

else

   {

      printf("( ");

      print\_optimal(i, s[i][j]);

      print\_optimal(s[i][j] + 1, j);

      printf(" )");

   }

}

void matmultiply()

{

long int q;

int k;

for(i=n;i>0;i--)

 {

   for(j=i;j<=n;j++)

    {

     if(i==j)

       m[i][j]=0;

     else

       {

        for(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;

          }

         }

        }

      }

 }

}

void main()

{

int k;

printf("Enter the no. of elements: ");

scanf("%d",&n);

for(i=1;i<=n;i++)

for(j=i+1;j<=n;j++)

{

 m[i][i]=0;

 m[i][j]=999999999;

 s[i][j]=0;

}

printf("\nThe randomly generated dimensions are: \n");

for(k=0;k<=n;k++)

{

 printf("P%d: ",k);

 int upper=46,lower=15;

 p[k]=(rand() % (upper-lower+1))+lower ;

 printf("%d\n",p[k]);

}

matmultiply();

printf("\nCost Matrix M:\n");

for(i=1;i<=n;i++)

 for(j=i;j<=n;j++)

  printf("m[%d][%d]: %ld\n",i,j,m[i][j]);

i=1,j=n;

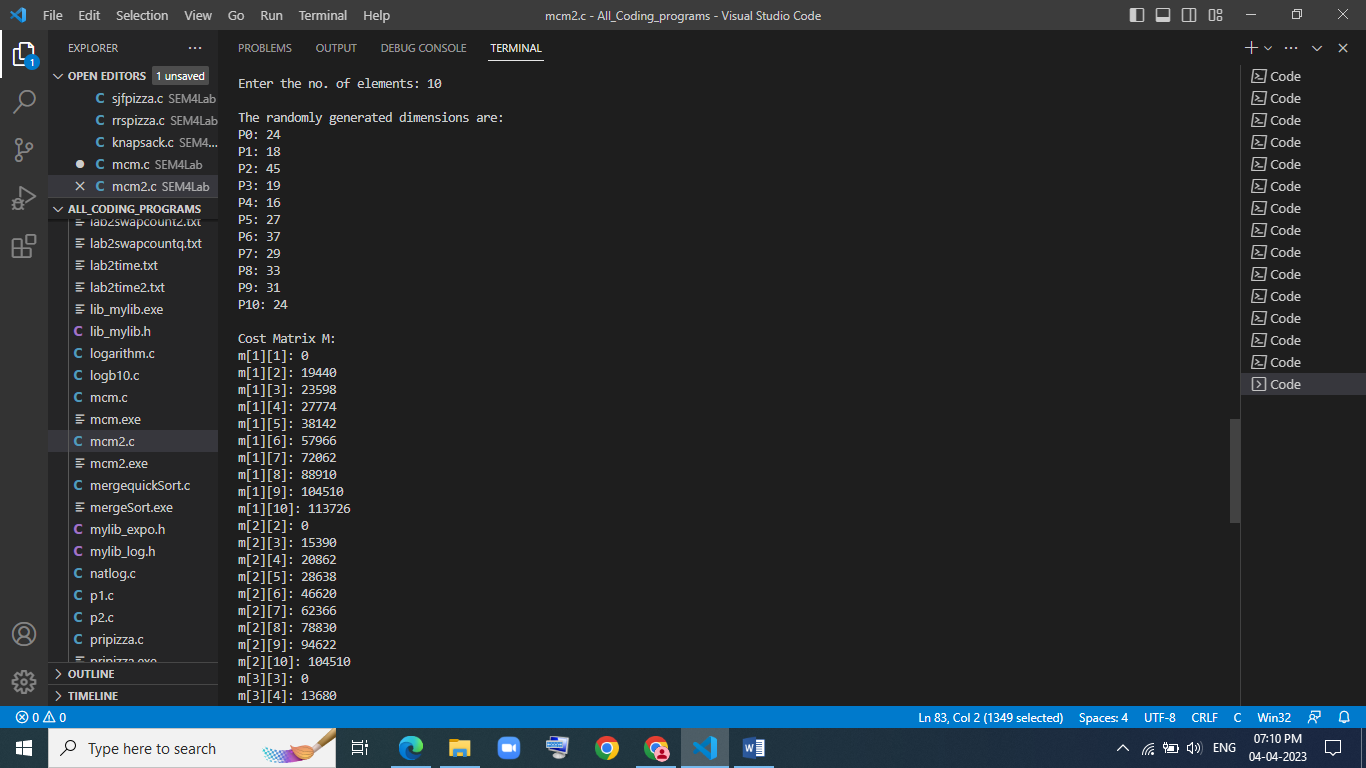
printf("\nMultiplication Sequence : ");

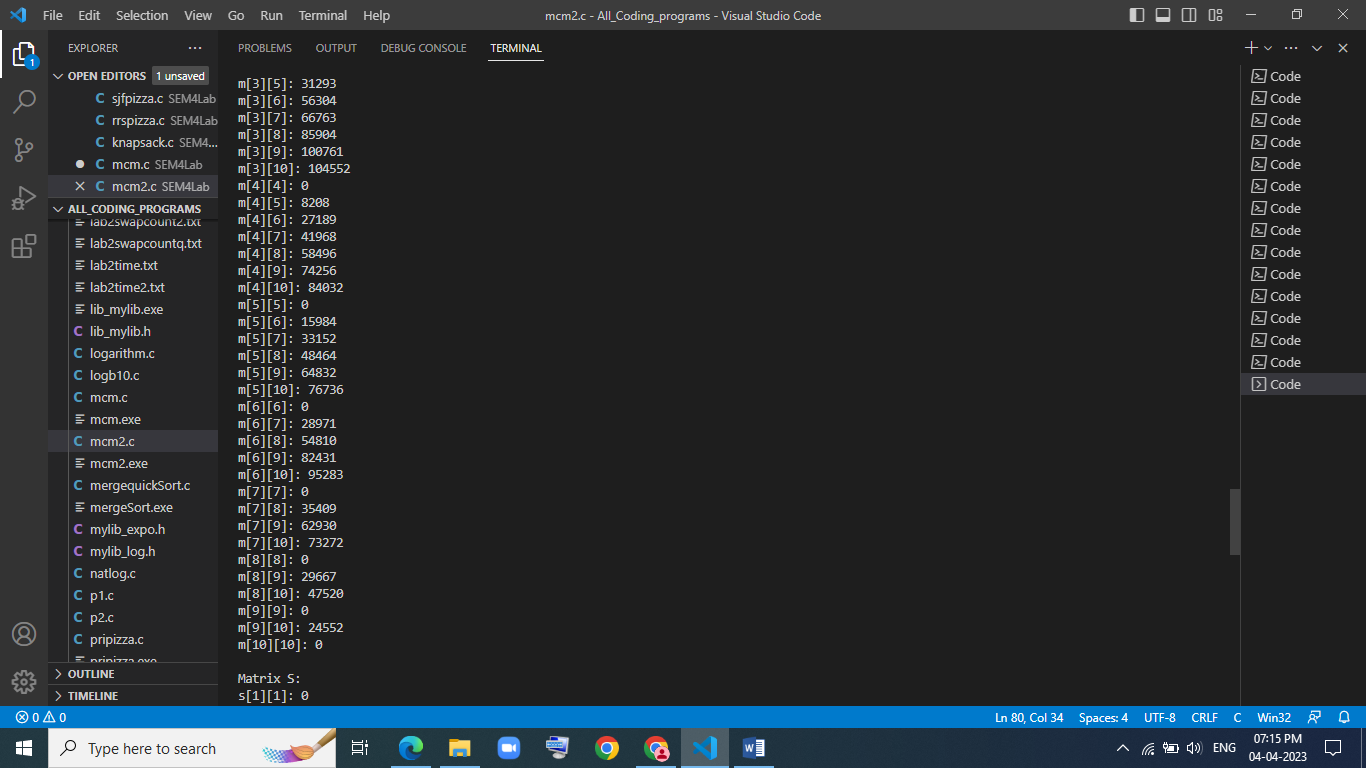
print\_optimal(i,j);

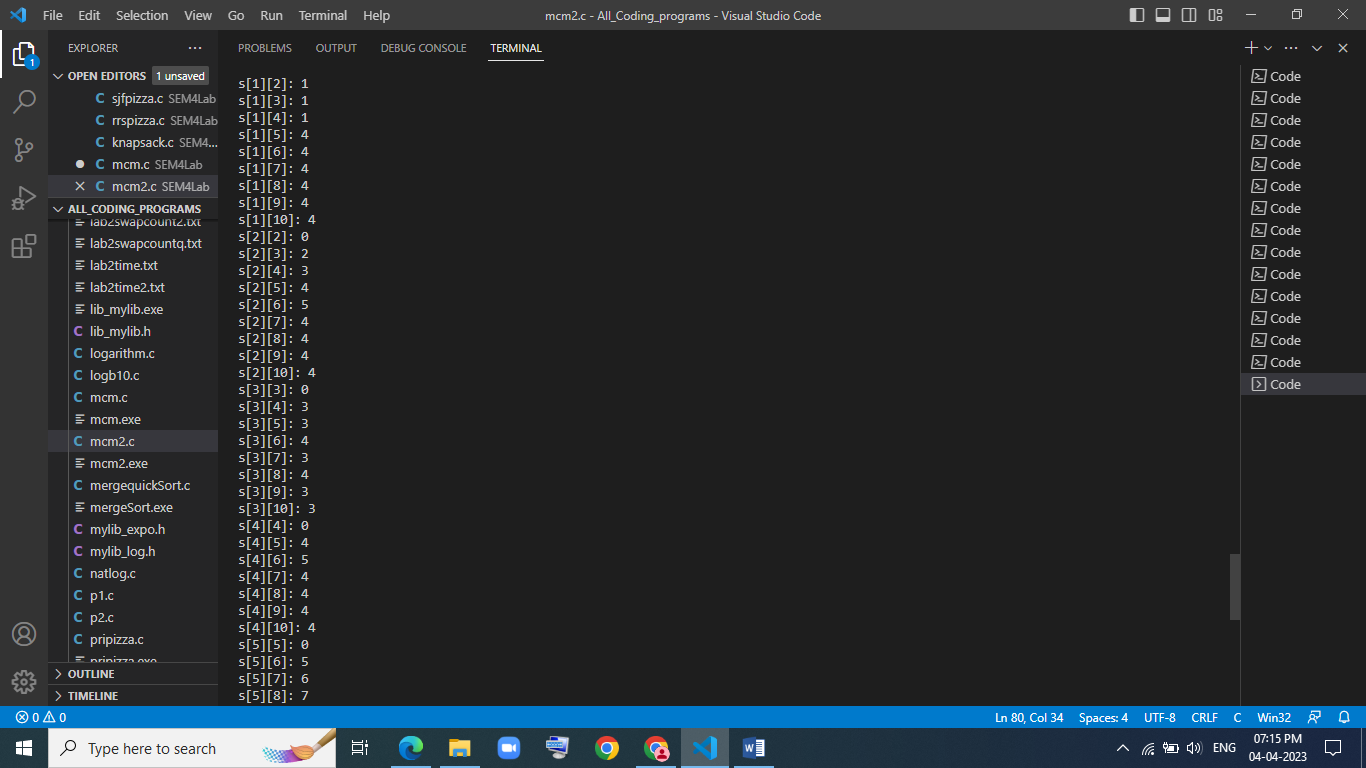
printf("\nMinimum number of multiplications is : %d ", m[1][n]);

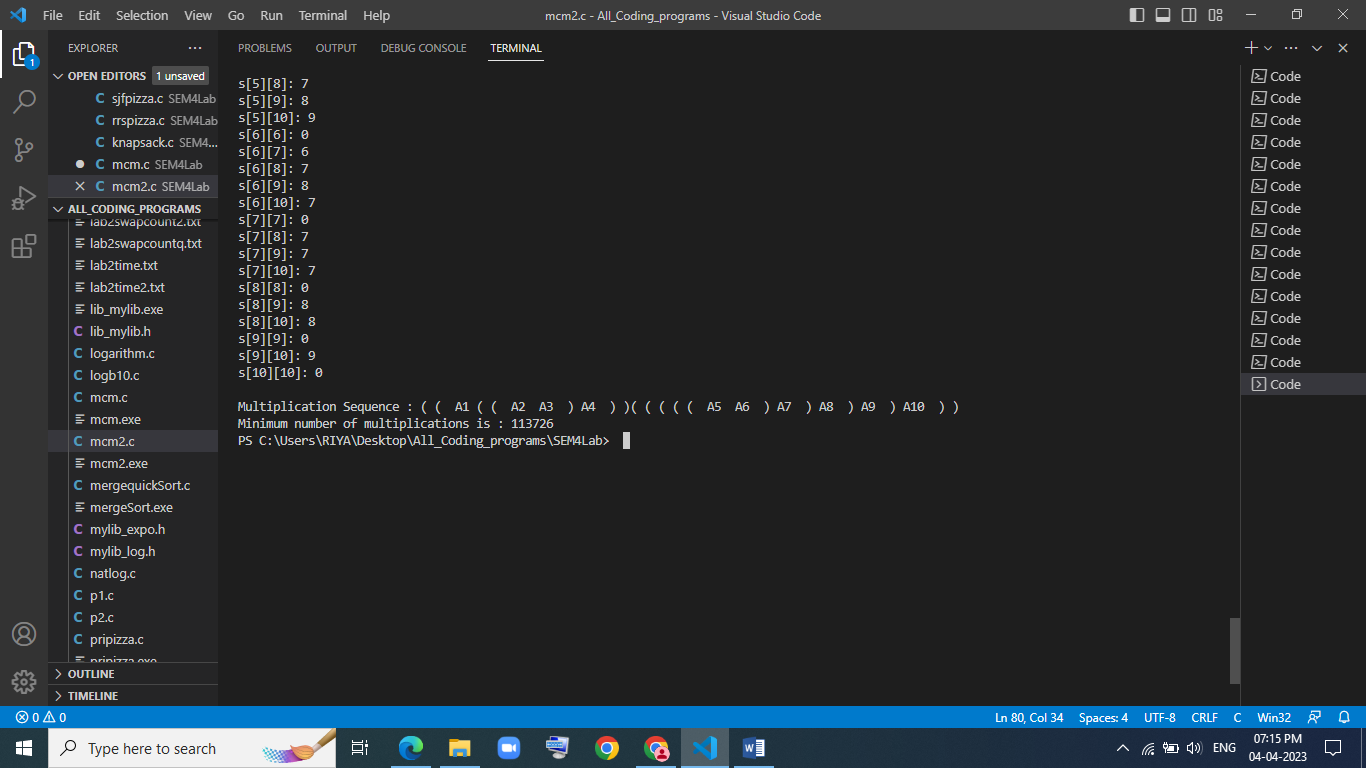
}

**Output and Observation:**



****

****

****

**The optimal solution of parenthesization has been found for the 10 randomly generated dimensions.**

**Conclusion:**

After performing the above experiment, I got to know how to perform matrix chain multiplication and find optimal paranthesization using dynamic programming approach.I could also find the minimum number of multiplications needed for the operation.