**ASSIGNMENT-9**

**EXPERIMENT – 16: Write a program to find an order in which given matrices are multiplied, so that total number of scalar multiplications are least.**

#include <limits.h>

#include <stdio.h>

int MatrixChainOrder(int p[], int i, int j)

{

if (i == j)

return 0;

int k;

int min = INT\_MAX;

int count;

for (k = i; k < j; k++)

{

count = MatrixChainOrder(p, i, k)

+ MatrixChainOrder(p, k + 1, j)

+ p[i - 1] \* p[k] \* p[j];

if (count < min)

min = count;

}

return min;

}

int main()

{

int arr[] = { 1, 2, 3, 4, 3 };

int n = sizeof(arr) / sizeof(arr[0]);

printf("Minimum number of multiplications is %d ",

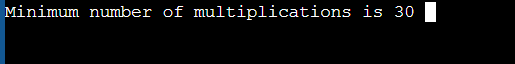
MatrixChainOrder(arr, 1, n - 1));

getchar();

return 0;

}

**OUTPUT-**

****

**EXPERIMENT – 17: Write a program to implement 0-1 Knapsack Problem using dynamic programming to get maximum total value in the knapsack.**

def knapSack(W, wt, val, n):

if n == 0 or W == 0 :

return 0

if (wt[n-1] > W):

return knapSack(W, wt, val, n-1)

else:

return max(

val[n-1] + knapSack(

W-wt[n-1], wt, val, n-1),

knapSack(W, wt, val, n-1))

if \_\_name\_\_ == '\_\_main\_\_':

val = [60, 100, 120]

wt = [10, 20, 30]

W = 50

n = len(val)

print ("Maximum Total Value=",knapSack(W, wt, val, n))

**OUTPUT-**

****