**Name**: Ravi Jamanbhai Makwana

**Roll No**: 21BCP418

Div-6, G-12

**Assignment-5**

**Aim:**

Implementation of Matrix Multiplication using standard matrix multiplication algorithm.

**Algorithm:**

matrixMultiply(A, B):

Assume dimension of A is (m x n), dimension of B is (p x q)

Begin

if n is not same as p, then exit

otherwise define C matrix as (m x q)

for i in range 0 to m - 1, do

for j in range 0 to q – 1, do

for k in range 0 to p, do

C[i, j] = C[i, j] + (A[i, k] \* A[k, j])

done

done

done

End

**Program:**

#include <stdio.h>

static int n;

void standard\_matrix\_mul(int arr1[n][n] , int arr2[n][n], int result\_arr[n][n], int n){

for (int i=0;i<n;i++){

for (int j=0;j<n;j++){

result\_arr[i][j]=0;

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

result\_arr[i][j] += arr1[i][k] \* arr2[k][j];

}

}

}

}

void display(int arr[n][n], int n){

for (int i=0;i<n;i++){

for (int j=0;j<n;j++){

printf("%d ",arr[i][j]);

}

printf("\n");

}

}

int main(){

printf("Enter size (n) of matrices: ");

scanf("%d",&n);

int arr1[n][n];

int arr2[n][n];

printf("Enter elements of Matrix-1:\n");

for (int i=0;i<n;i++){

for (int j=0;j<n;j++){

printf("Enter element [%d][%d]: ",(i+1),(j+1));

int temp;

scanf("%d",&temp);

arr1[i][j] = temp;

}

}

printf("\nEnter elements of Matrix-2:\n");

for (int i=0;i<n;i++){

for (int j=0;j<n;j++){

printf("Enter element [%d][%d]: ",(i+1),(j+1));

int temp;

scanf("%d",&temp);

arr2[i][j] = temp;

}

}

printf("\nMatrix-1:\n");

display(arr1,n);

printf("\nMatrix-2:\n");

display(arr2,n);

int resultArr[n][n];

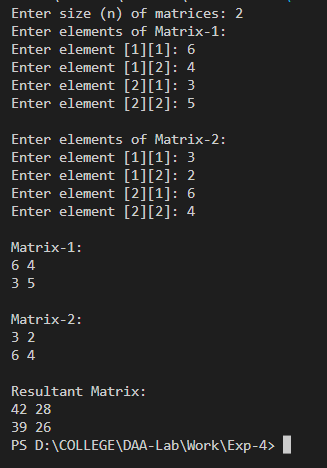
standard\_matrix\_mul(arr1,arr2,resultArr,n);

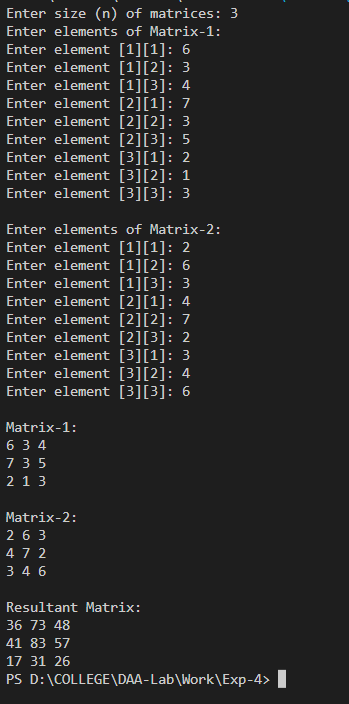
printf("\nResultant Matrix:\n");

display(resultArr,n);

}

**Output:**

****



**Analysis:**

**Time Complexity Analysis:**

The naive matrix multiplication algorithm contains three nested loops. For each iteration of the outer loop, the total number of the runs in the inner loops would be equivalent to the length of the matrix. Here, integer operations take O(n) time. In general, if the length of the matrix is n, the total time complexity would be:

**O(n\*n\*n) = O(n3).**