**PROBLEM 3(A):**

Write a program in C language to multiply two square matrices using the **iterative approach**. Compare the execution time for different matrix sizes.

**SOLUTION:**

**C-CODE:**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

void generateRandomMatrix(int n, int mat[n][n]){

    for(int i = 0; i < n; i++){

        for(int j = 0; j < n; j++){

            mat[i][j] = rand()%10;

        }

    }

}

int matrix\_multiplication(int n,  int mat1[n][n], int mat2[n][n]){

    int result[n][n];

    for(int i = 0; i < n; i++){

        for(int j = 0; j < n; j++){

            result[i][j] = 0;

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

                result[i][j] += mat1[i][k] \* mat2[k][j];

            }

        }

    }

    printf("Resultant Matrix:\n");

    for(int i = 0; i < n; i++){

        for(int j = 0; j < n; j++){

            printf("%d\t", result[i][j]);

        }

        printf("\n");

    }

    return 0;

}

int main(){

    int n;

    printf("Enter the size of matrix: ");

    scanf("%d", &n);

    int mat1[n][n], mat2[n][n], result[n][n];

    generateRandomMatrix(n, mat1);

    generateRandomMatrix(n , mat2);

    printf("\n");

    printf("Matrix 1 : \n");

    for(int i = 0; i < n; i++) {

        for(int j = 0; j <n; j++){

            printf("%d\t", mat1[i][j]);

        }

        printf("\n");

    }

    printf("\n");

    printf("Matrix 2 : \n");

    for(int i = 0; i < n; i++) {

        for(int j = 0; j < n; j++){

            printf("%d\t", mat2[i][j]);

        }

        printf("\n");

    }

    printf("\n");

    clock\_t start = clock();

    matrix\_multiplication(n, mat1, mat2);

    clock\_t end = clock();

    printf("\n");

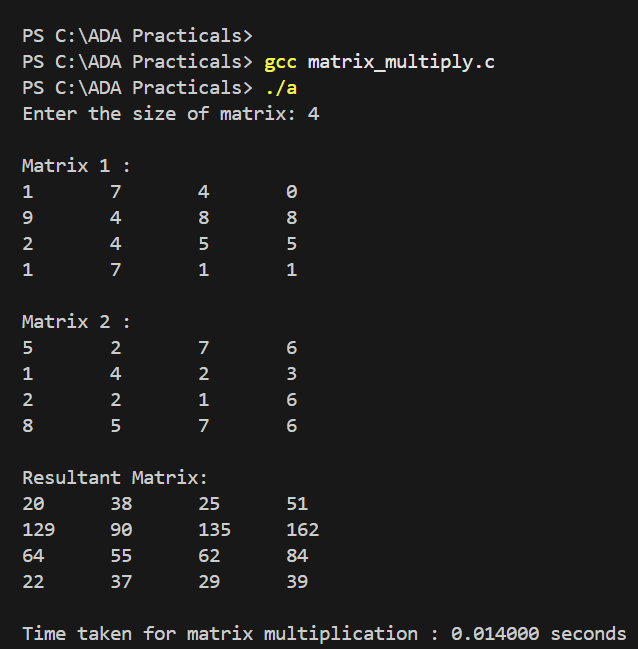
    double time\_taken = ((double)(end-start))/CLOCKS\_PER\_SEC;

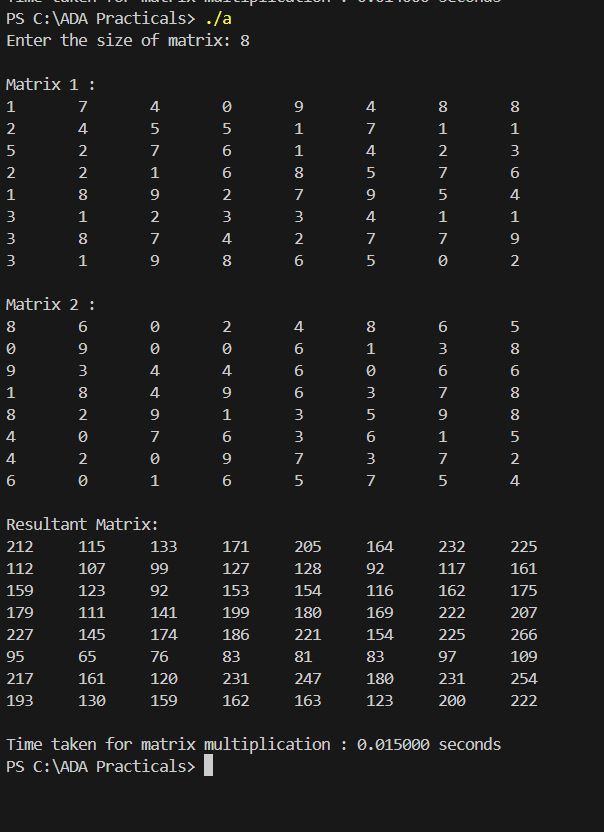
    printf("Time taken for matrix multiplication : %f seconds\n", time\_taken);

    return 0;

}

**OUTPUT:**





**PROBLEM 3(B):**

Write a program in C language to multiply two square matrices using the . Compare the execution time for different matrix sizes.

**SOLUTION:**

**C-CODE:**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

void generateRandomMatrix(int n, int mat[n][n]){

    for(int i = 0; i < n; i++){

        for(int j = 0; j < n; j++){

            mat[i][j] = rand()%10;

        }

    }

}

int matrix\_multiplication(int n,  int mat1[n][n], int mat2[n][n]){

    int result[n][n];

    for(int i = 0; i < n; i++){

        for(int j = 0; j < n; j++){

            result[i][j] = 0;

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

                result[i][j] += mat1[i][k] \* mat2[k][j];

            }

        }

    }

    printf("Resultant Matrix:\n");

    for(int i = 0; i < n; i++){

        for(int j = 0; j < n; j++){

            printf("%d\t", result[i][j]);

        }

        printf("\n");

    }

    return 0;

}

int main(){

    int n;

    printf("Enter the size of matrix: ");

    scanf("%d", &n);

    int mat1[n][n], mat2[n][n], result[n][n];

    generateRandomMatrix(n, mat1);

    generateRandomMatrix(n , mat2);

    printf("\n");

    printf("Matrix 1 : \n");

    for(int i = 0; i < n; i++) {

        for(int j = 0; j <n; j++){

            printf("%d\t", mat1[i][j]);

        }

        printf("\n");

    }

    printf("\n");

    printf("Matrix 2 : \n");

    for(int i = 0; i < n; i++) {

        for(int j = 0; j < n; j++){

            printf("%d\t", mat2[i][j]);

        }

        printf("\n");

    }

    printf("\n");

    clock\_t start = clock();

    matrix\_multiplication(n, mat1, mat2);

    clock\_t end = clock();

    printf("\n");

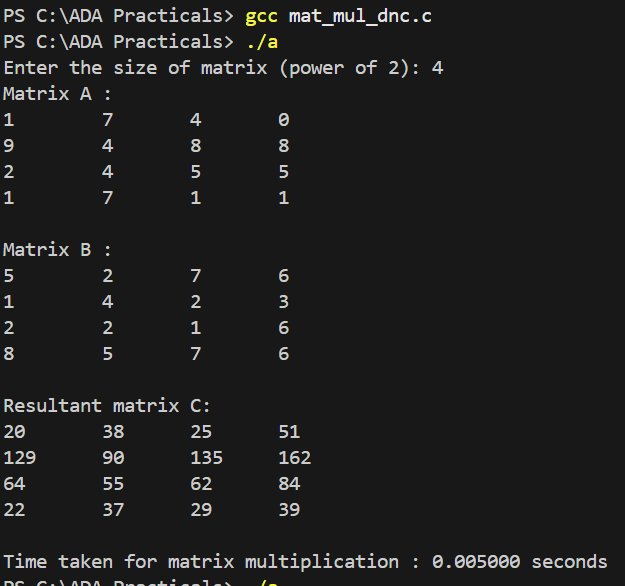
    double time\_taken = ((double)(end-start))/CLOCKS\_PER\_SEC;

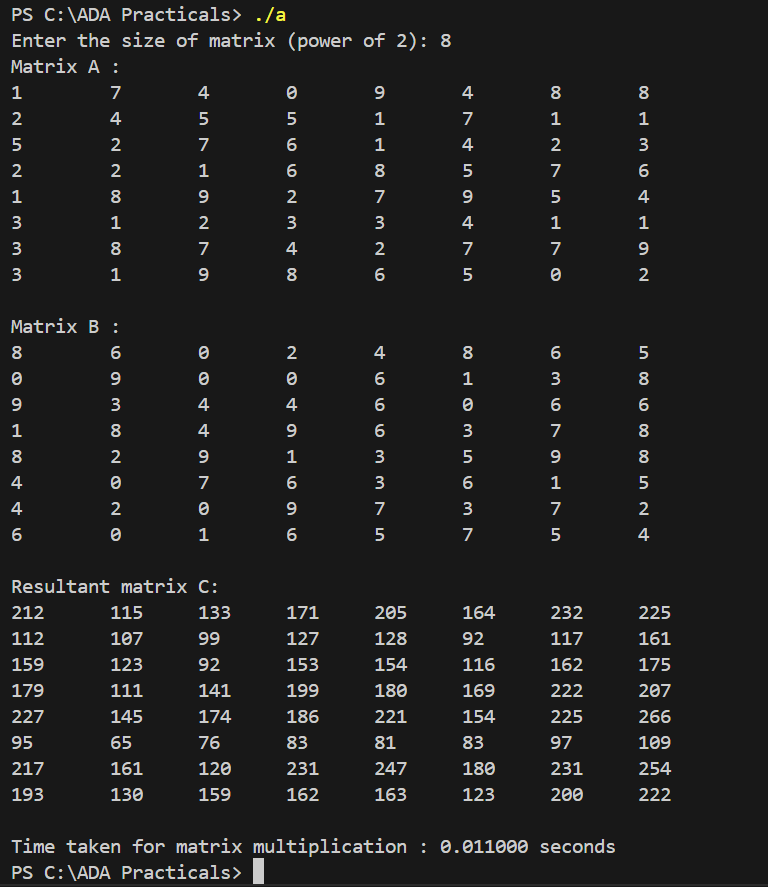
    printf("Time taken for matrix multiplication : %f seconds\n", time\_taken);

    return 0;

}

***OUTPUT:***



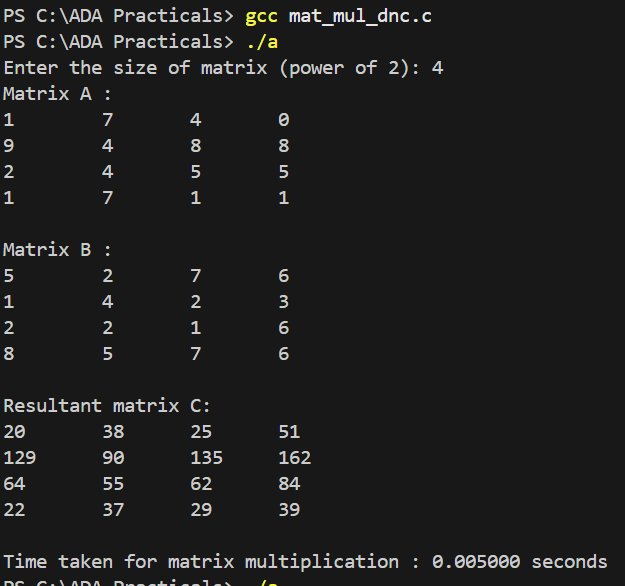


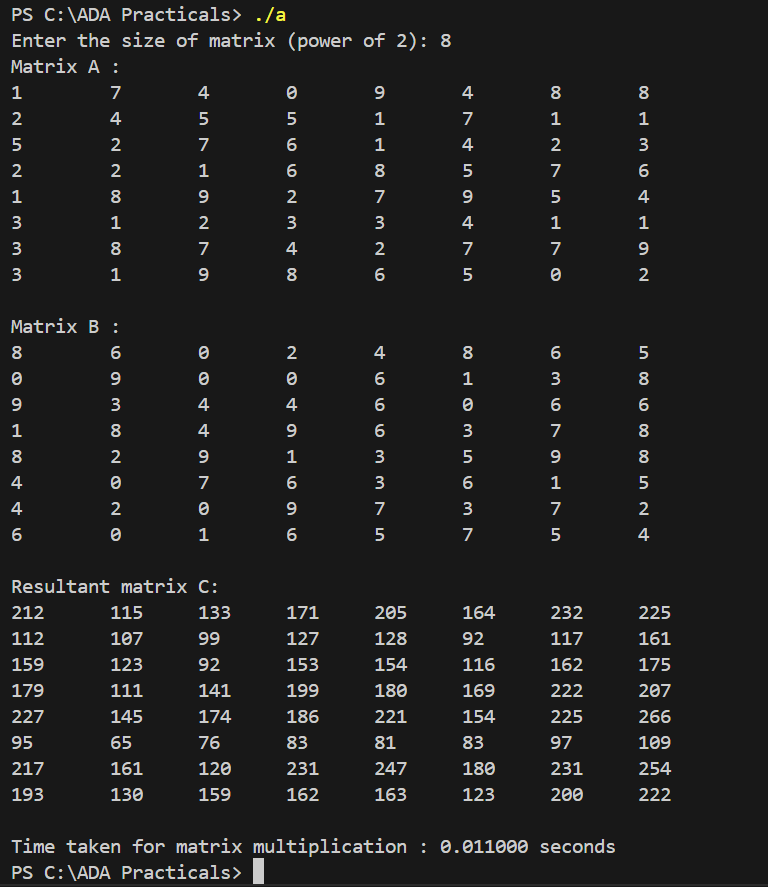
**PROBLEM 3(C):**

Given two square matrices A and B of size n × n (n is a power of 2), write a C code to multiply them using, which reduces the number of recursive multiplications from 8 to 7 by introducing additional addition/subtraction operations. Compare the execution time for different matrix sizes.

**SOLUTION:**

***OUTPUT:***





**PROBLEM 3(C):**

Given two square matrices A and B of size n × n (n is a power of 2), write a C code to multiply them using, which reduces the number of recursive multiplications from 8 to 7 by introducing additional addition/subtraction operations. Compare the execution time for different matrix sizes.

**SOLUTION:**

***C-CODE:***

***C-CODE:***

#include <stdio.h>

#include <stdlib.h>

#include<time.h>

void generateRandomMatrix(int n, int mat[n][n]){

    for(int i = 0; i < n; i++){

        for(int j = 0; j < n; j++){

            mat[i][j] = rand()%10;

        }

    }

}

void add(int n, int A[n][n], int B[n][n], int C[n][n]) {

    for (int i = 0; i < n; i++)

        for (int j = 0; j < n; j++)

            C[i][j] = A[i][j] + B[i][j];

}

void subtract(int n, int A[n][n], int B[n][n], int C[n][n]) {

    for (int i = 0; i < n; i++)

        for (int j = 0; j < n; j++)

            C[i][j] = A[i][j] - B[i][j];

}

void multiply(int n, int A[n][n], int B[n][n], int C[n][n]) {

    if (n == 2) {

        C[0][0] = A[0][0]\*B[0][0] + A[0][1]\*B[1][0];

        C[0][1] = A[0][0]\*B[0][1] + A[0][1]\*B[1][1];

        C[1][0] = A[1][0]\*B[0][0] + A[1][1]\*B[1][0];

        C[1][1] = A[1][0]\*B[0][1] + A[1][1]\*B[1][1];

        return;

    }

    int k = n/2;

    int A11[k][k], A12[k][k], A21[k][k], A22[k][k];

    int B11[k][k], B12[k][k], B21[k][k], B22[k][k];

    int C11[k][k], C12[k][k], C21[k][k], C22[k][k];

    int M1[k][k], M2[k][k];

    int P[k][k], Q[k][k], R[k][k], S[k][k], T[k][k], U[k][k], V[k][k];

    for (int i=0; i<k; i++)

        for (int j=0; j<k; j++)

            C11[i][j]=C12[i][j]=C21[i][j]=C22[i][j]=0;

    // Divide A and B into submatrices

    for (int i=0; i<k; i++) {

        for (int j=0; j<k; j++) {

            A11[i][j] = A[i][j];

            A12[i][j] = A[i][j+k];

            A21[i][j] = A[i+k][j];

            A22[i][j] = A[i+k][j+k];

            B11[i][j] = B[i][j];

            B12[i][j] = B[i][j+k];

            B21[i][j] = B[i+k][j];

            B22[i][j] = B[i+k][j+k];

        }

    }

    // P = (A11+A22)\*(B11+B22)

    add(k, A11, A22, M1);

    add(k, B11, B22, M2);

    multiply(k, M1, M2, P);

    // Q = (A21+A22)\*B11

    add(k, A21, A22, M1);

    multiply(k, M1, B11, Q);

    // R= A11\*(B12-B22)

    subtract(k, B12, B22, M1);

    multiply(k, A11, M1, R);

    // S = A22\*(B21-B11)

    subtract(k, B21, B11, M1);

    multiply(k, A22, M1, S);

    // T = (A11+A12)\*B22

    add(k, A11, A12, M1);

    multiply(k, M1, B22, T);

    // U = (A21-A11)\*(B11+B12)

    subtract(k, A21, A11, M1);

    add(k, B11, B12, M2);

    multiply(k, M1, M2, U);

    // V = (A12-A22)\*(B21+B22)

    subtract(k, A12, A22, M1);

    add(k, B21, B22, M2);

    multiply(k, M1, M2, V);

    // C11 = P+S-T+V = (P+V)+(S-T)

    add(k, P, V, M1);

    subtract(k, S, T, M2);

    add(k, M1, M2, C11);

    // C12 = R+T

    add(k, R, T, C12);

    // C21 = Q+S

    add(k, Q, S, C21);

    // C22 = P+R-Q+U = (P+U)+(R-Q)

    add(k, P, U, M1);

    subtract(k, R, Q, M2);

    add(k, M1, M2, C22);

    // Assemble result matrix C from submatrices

    for (int i=0; i<k; i++) {

        for (int j=0; j<k; j++) {

            C[i][j] = C11[i][j];

            C[i][j+k] = C12[i][j];

            C[i+k][j] = C21[i][j];

            C[i+k][j+k] = C22[i][j];

        }

    }

}

void printMatrix(int n, int M[n][n]) {

    for (int i=0; i<n; i++) {

        for (int j=0; j<n; j++)

            printf("%d\t", M[i][j]);

        printf("\n");

    }

}

int main() {

    int n;

    printf("Enter the size of matrix (power of 2): ");

    scanf("%d", &n);

    int A[n][n], B[n][n], C[n][n];

    generateRandomMatrix(n, A);

    printf("Matrix A : \n");

    printMatrix(n,A);

    printf("\n");

    generateRandomMatrix(n, B);

    printf("Matrix B : \n");

    printMatrix(n,B);

    printf("\n");

    clock\_t start = clock();

    multiply(n, A, B, C);

    printf("Resultant matrix C:\n");

    printMatrix(n, C);

    clock\_t end = clock();

    printf("\n");

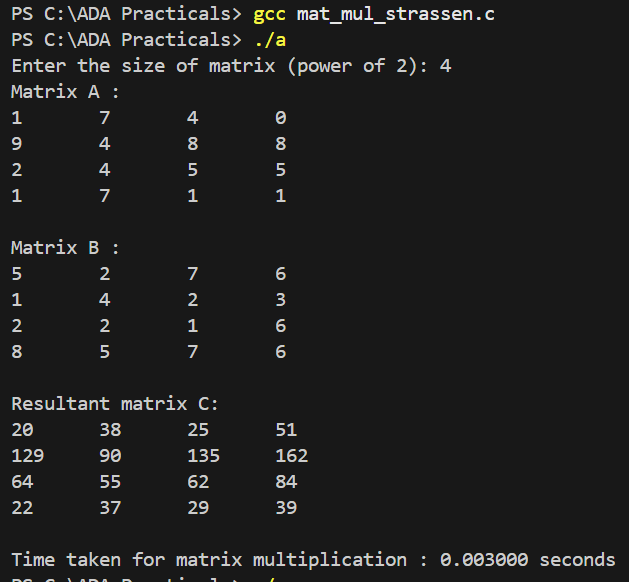
    double time\_taken = ((double)(end-start))/CLOCKS\_PER\_SEC;

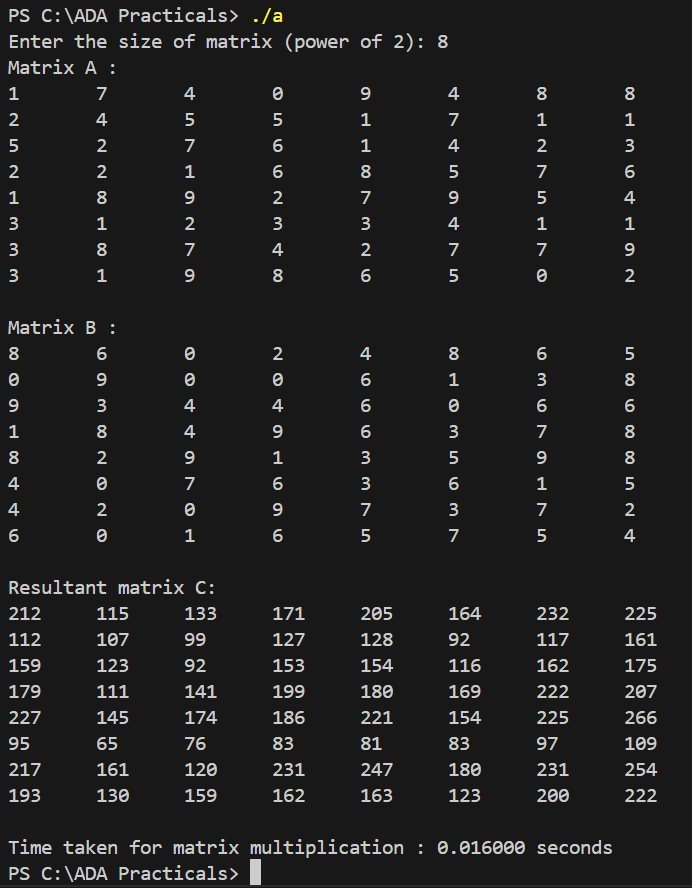
    printf("Time taken for matrix multiplication : %f seconds\n", time\_taken);

    return 0;

}

***OUTPUT:***





***OUTPUT:***

