Code:

#include<stdio.h>

#include<stdlib.h>

#include<omp.h>

#define N 4

void matrixCreation(int A[][N], int B[][N]){ for (int i = 0; i<N; i++){ for (int j = 0; j< N; j++){

A[i][j] = i\*j;

B[i][j] = i\*j;

}

}

}

void matrixMultiplication(int A[][N], int B[][N], int C[][N], int chunk\_size){

#pragma omp parallel for schedule (static, chunk\_size)

for(int i = 0; i<N; i++){ for(int j = 0; j<N; j++){

C[i][j] = 0;

for(int k = 0; k<N; k++){ C[i][j] += A[i][k] \* B[k][j];

}

}

}

}

void display\_output(int C[][N]){ int tid = omp\_get\_thread\_num(); printf("Output matrix from thread %d:\n", tid); for(int i = 0; i<N; i++){ for(int j = 0; j<N; j++){

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

}

printf("\n");

}

printf("Thread ID of the displayed thread: %d\n", tid);

} int main(){

int A[N][N], B[N][N], C[N][N]; int P, m;

printf("Enter the number of threads(P): "); scanf("%d", &P);

if(P<=5){

printf("P must be greater than 5 for parallel execution\n"); return 1;

}

printf("Enter chunk size(m): "); scanf("%d", &m);

//initialise matrices matrixCreation(A,B); //matrix Multiplication

#pragma omp parallel num\_threads(P) if (P > 5) default(none) shared(A,B,C,m)

{

matrixMultiplication(A,B,C,m);

//display matrix

#pragma omp single

{

display\_output(C);

}

}

return 0;

}

Output:



