Kodi minimal per C – Matrix Multiplication with MPI

```
#include <stdio.h> <stdlib.h> <mpi.h>
#define N 4  // Matrix size
int main(int argc, char** argv) {
  int rank, size;
  int A[N][N], B[N][N], C[N][N];
  int local_A[N/2][N], local_C[N/2][N];  // Assuming 2 processes

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  int rows_per_process = N / size;

  // Process 0 initializes matrices
  if (rank == 0) {
 // Initialize matrix A dhe Matrica B

  // Broadcast matrix B to all processes
  MPI_Bcast(B, N*N, MPI_INT, 0, MPI_COMM_WORLD);

  // Scatter rows of matrix A to all processe
MPI_Scatter(A, rows_per_process*N, MPI_INT,  local_A, rows_per_process*N, MPI_INT, 0,
MPI_COMM_WORLD);

  // Each process computes its portion of matrix C
  for (int i = 0; i < rows_per_process; i++) {
    for (int j = 0; j < N; j++) {
      local_C[i][j] = 0;
      for (int k = 0; k < N; k++) {
        local_C[i][j] += local_A[i][k] * B[k][j];
      } } }

  // Gather results back to process 0
  MPI_Gather(local_C, rows_per_process*N, MPI_INT,
        C, rows_per_process*N, MPI_INT, 0, MPI_COMM_WORLD);

  // Process 0 prints the result
  if (rank == 0) {
    printf("\nMatrix C (A * B):\n");
        MPI_Finalize();  -> return 0;
```

Kodi minimal ne C – Shumatorja e te gjithe elementeve te nje matrice

```
#include <stdio.h>
#include <mpi.h>
#define N 10

int main(int argc, char**argv){
   int rank, size;
   int A[N][N];
   int local_A[2][N];  // 5 procese = 2 rreshta/proces
   int localSum = 0, totalSum = 0;

   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);

   // Process 0 inicializon
   if(rank == 0){ }

   // Scatter
   MPI_Scatter(A, 2*N, MPI_INT, local_A, 2*N, MPI_INT, 0, MPI_COMM_WORLD);

   // Local sum
   for(int i = 0; i < 2; i++)
      for(int j = 0; j < N; j++)
         localSum += local_A[i][j];

   // Reduce
   MPI_Reduce(&localSum, &totalSum, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

   if(rank == 0)
      printf("Total sum: %d\n", totalSum);

   MPI_Finalize();
   return 0;
}
```