```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>  //I have installed the mpi library
//I have used this command 'sudo apt-get install mpich libmpich-dev' (ubuntu)

#define generate_data(i,j) (i)+(j)*(j)

int main(int argc, char **argv) {
    int i, j, pid, np, mtag, count;
    double t0, t1;
    int data[100][100], row_sum[100];
    MPI_Status status;
    MPI_Request req_s, req_r[2]; // Requests for non-blocking reception
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &pid);
    MPI_Comm_size(MPI_COMM_WORLD, &np);

    if (pid == 0) {
        // Process 0: Generate and send data
        for (i = 0; i < 50; i++) {
            for (j = 0; j < 100; j++) {
                data[i][j] = generate_data(i, j);
            }
        }
        mtag = 1;
        MPI_Isend(data, 5000, MPI_INT, 1, mtag, MPI_COMM_WORLD, &req_s);
        for (i = 50; i < 100; i++) {
            for (j = 0; j < 100; j++) {
                data[i][j] = generate_data(i, j);
            }
        }
        for (i = 50; i < 100; i++) {
            row_sum[i] = 0;
            for (j = 0; j < 100; j++) {
                row_sum[i] += data[i][j];
            }
        }
        MPI_Wait(&req_s, &status);
        // Receive computed row sums from pid 1
        mtag = 2;
        MPI_Recv(row_sum, 50, MPI_INT, 1, mtag, MPI_COMM_WORLD, &status);
        for (i = 0; i < 100; i++) {
            printf(" %d ", row_sum[i]);
            if (i % 10 == 9) printf("\n");
        }
    } else { // pid == 1
        // Process 1: Receive and compute row sums in chunks
        mtag = 1;

        // First receive 25 rows non-blocking
        MPI_Irecv(&data[0][0], 2500, MPI_INT, 0, mtag, MPI_COMM_WORLD, &req_r[0]);

        // While receiving first 25 rows, compute sums for the other 25 rows
        for (i = 25; i < 50; i++) {
            for (j = 0; j < 100; j++) {
                data[i][j] = generate_data(i, j); // Initialize the second half
            }
            row_sum[i] = 0;
            for (j = 0; j < 100; j++) {
```

```
                row_sum[i] += data[i][j];
            }
        }

        // Wait for the first 25 rows to be received
        MPI_Wait(&req_r[0], &status);

        // Start computing row sums for the first 25 rows
        for (i = 0; i < 25; i++) {
            row_sum[i] = 0;
            for (j = 0; j < 100; j++) {
                row_sum[i] += data[i][j];
            }
        }

        // Send computed row sums to pid 0
        mtag = 2;
        MPI_Send(row_sum, 50, MPI_INT, 0, mtag, MPI_COMM_WORLD);
    }

    MPI_Finalize();
    return 1;
}
```