CS 5379 Project 1. Due: Sep 12 before midnight.

The attached code computes row sums of matrix data[100][100] by 2 processes.

In this assignment, please rewrite the highlighted code segment so that on pid==1, there is significant overlapping of communication with computation.

Since the requirement is that there is significant overlapping of communication with computation on pid==1, so do not re-assign the sub-computations between the two processes, i.e. the process with pid==1 does only the row sums for row 50 to row 99, where the data needed by the row sums to be done by the process with pid==1 are to be received from the process with pid==0. This assignment asks you to achieve overlapping of receiving of some rows (some of row 50 to row 99) with the computation row sums for some rows (some of row 50 to row 99).

Grading criteria:
1. 40%, Write a short description of your technical idea (no more than 1 page)
2. 50%, correct and clear C code implementation of your technical idea
3. 10%, clear code comments to help grader understand your code.

Clarity/understandability of description and code:
If grader cannot understand your description of your technical idea, you'll lose 40% from item 1. If TA cannot understand your code, you'll lose 60% from items 2 & 3.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <mpi.h>

#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;


  MPI_Init( &argc, &argv );
  MPI_Comm_rank(MPI_COMM_WORLD, &pid);
  MPI_Comm_size(MPI_COMM_WORLD, &np);
  if(pid == 0) { // generate 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%5 ==0) printf("\n");
    }
  }
  else { /*** pid == 1 ***/
    mtag = 1 ;
    MPI_Recv(data, 5000, MPI_INT, 0, mtag, MPI_COMM_WORLD, &status) ;

    for(i=0; i<50; 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;
} /***************** End of function main() *******************/
```