**20BCE1550**

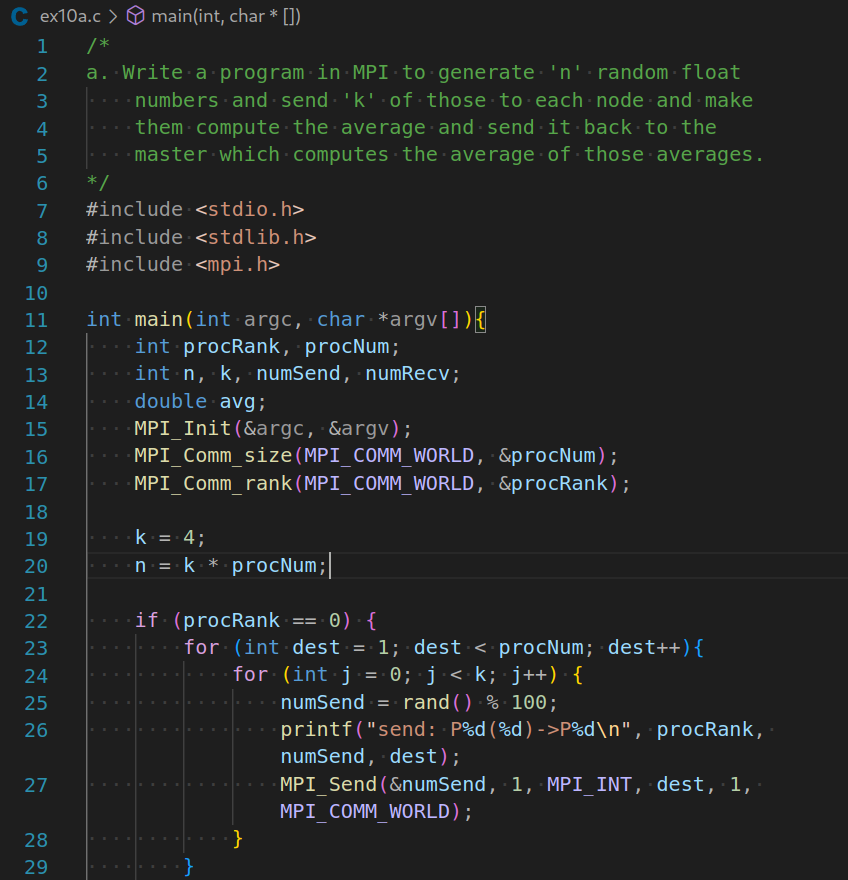
**Samridh Anand Paatni**

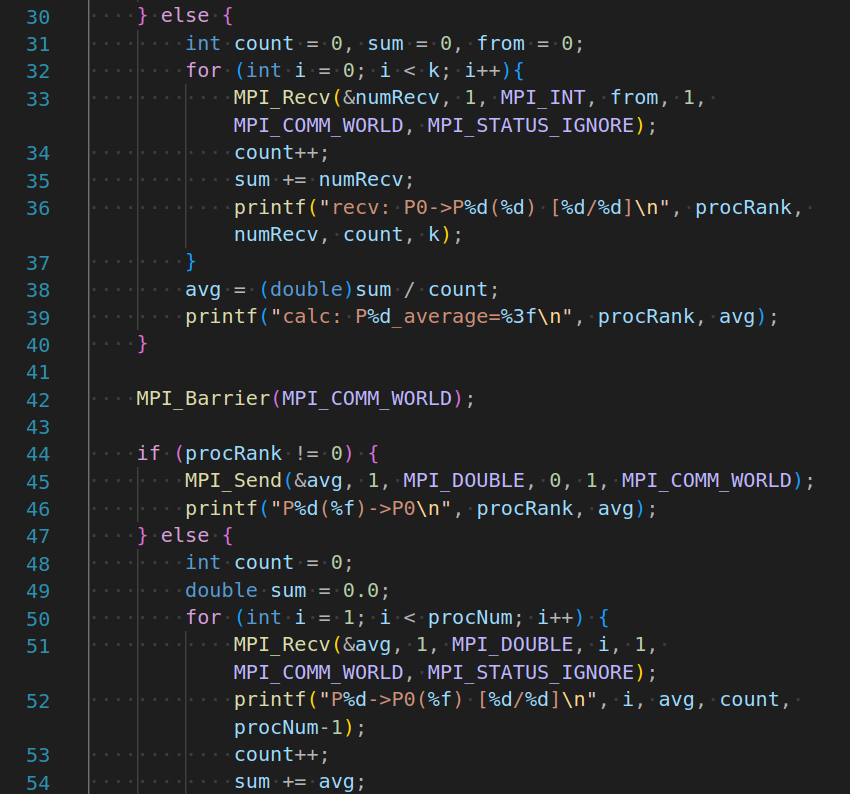
**CSE4001 Lab 10**

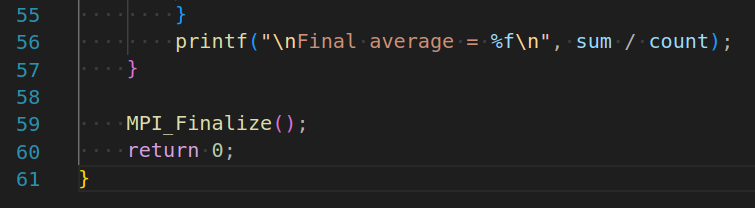
**MPI Collective Communication**

**a:**

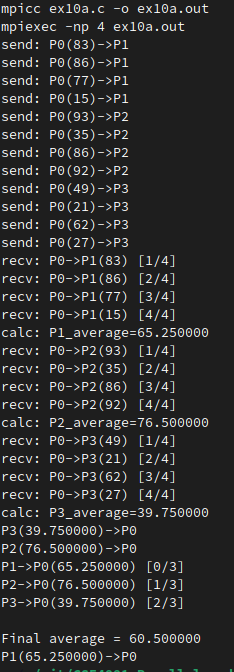
**Code:**





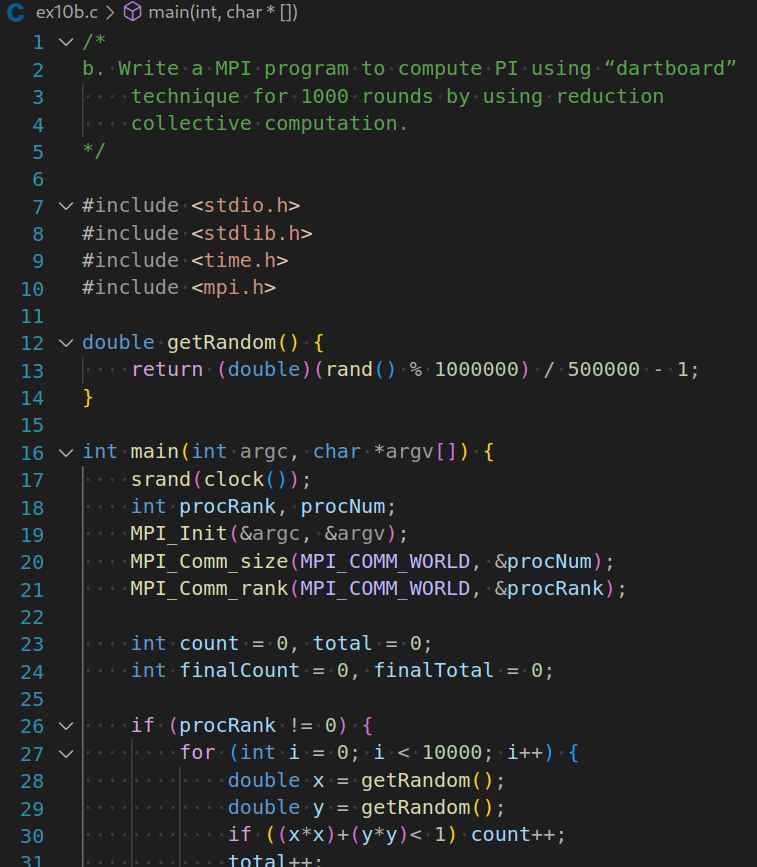


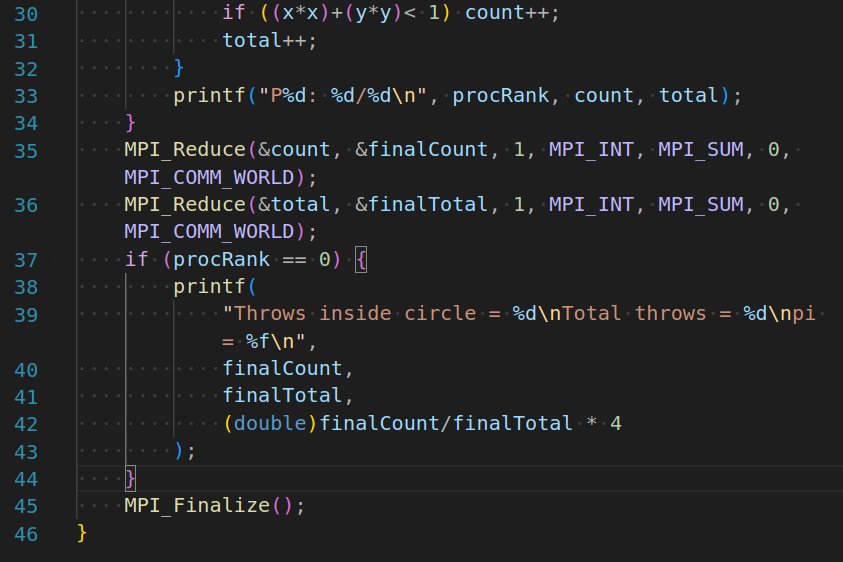
**Output:**



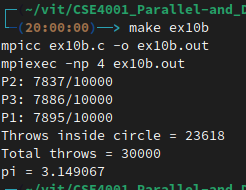
**b:**

**Code:**

****



**Output:**



**c:**

**Code:**

**/\***

c. Write a MPI program to perform matrix multiplication

(1000x1000) using scatter and gather routines.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <mpi.h>

#include <mpi.h>

#include <stdio.h>

#define SIZE 8

int A[SIZE][SIZE], B[SIZE][SIZE], C[SIZE][SIZE];

void fill\_matrix(int m[SIZE][SIZE])

{

static int n=0;

int i, j;

for (i=0; i<SIZE; i++)

for (j=0; j<SIZE; j++)

m[i][j] = n++;

}

void print\_matrix(int m[SIZE][SIZE])

{

int i, j = 0;

for (i=0; i<SIZE; i++) {

printf("\n\t| ");

for (j=0; j<SIZE; j++)

printf("%2d ", m[i][j]);

printf("|");

}

}

int main(int argc, char \*argv[])

{

int myrank, P, from, to, i, j, k;

int tag = 666; /\* any value will do \*/

MPI\_Status status;

MPI\_Init (&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myrank); /\* who am i \*/

MPI\_Comm\_size(MPI\_COMM\_WORLD, &P); /\* number of processors \*/

/\* Just to use the simple variants of MPI\_Gather and MPI\_Scatter we \*/

/\* impose that SIZE is divisible by P. By using the vector versions, \*/

/\* (MPI\_Gatherv and MPI\_Scatterv) it is easy to drop this restriction. \*/

if (SIZE%P!=0) {

if (myrank==0) printf("Matrix size not divisible by number of processors\n");

MPI\_Finalize();

exit(-1);

}

from = myrank \* SIZE/P;

to = (myrank+1) \* SIZE/P;

/\* Process 0 fills the input matrices and broadcasts them to the rest \*/

/\* (actually, only the relevant stripe of A is sent to each process) \*/

if (myrank==0) {

fill\_matrix(A);

fill\_matrix(B);

}

MPI\_Bcast (B, SIZE\*SIZE, MPI\_INT, 0, MPI\_COMM\_WORLD);

MPI\_Scatter (A, SIZE\*SIZE/P, MPI\_INT, A[from], SIZE\*SIZE/P, MPI\_INT, 0, MPI\_COMM\_WORLD);

printf("computing slice %d (from row %d to %d)\n", myrank, from, to-1);

for (i=from; i<to; i++)

for (j=0; j<SIZE; j++) {

C[i][j]=0;

for (k=0; k<SIZE; k++)

C[i][j] += A[i][k]\*B[k][j];

}

MPI\_Gather (C[from], SIZE\*SIZE/P, MPI\_INT, C, SIZE\*SIZE/P, MPI\_INT, 0, MPI\_COMM\_WORLD);

if (myrank==0) {

printf("\n\n");

print\_matrix(A);

printf("\n\n\t \* \n");

print\_matrix(B);

printf("\n\n\t = \n");

print\_matrix(C);

printf("\n\n");

}

MPI\_Finalize();

return 0;

}

**Output:**

