//NAMES: DivyaShree H B

//required header files

#include <stdio.h>

#include <stdlib.h>

#include <mpi.h>

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function : total\_sum()

// Parameters: int \* array, int numItems

// It calculates sum by adding items in the array

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int total\_sum(int \* array, int numItems){

int tSum=0,i;

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

tSum=tSum+array[i];

}

return tSum;

}

void main(){

int my\_rank, comm\_sz, my\_id; // MPI variables

int number = 32; //number to which summation is calculated

double start, finish; // variables for calculating time

int i,j,k,l,m,p;

int count; //number of values assigned to each processor

// //other required variables, such as loop control vars

int recv\_sum; //sum received from slave processors

int master\_sum = 0; //sum calculated by master process

int partial\_sum = 0; //sum calculated by slave processors

int final\_sum; //store final sum

//Int temp\_sum=0;

//3 required MPI functions

MPI\_Init(NULL, NULL);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &comm\_sz);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &my\_rank);

//determine how many numbers should be assigned to each process

count = (number/comm\_sz);

//master process is 0

if (my\_rank == 0) {

//create array to store numbers

int \* array = (int \*)malloc(sizeof(int)\* number);

//store values in the array from 1 to declared number

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

array[i]=i+1;

}

//create array to store partial sum calculated by each process

int \* summation = (int \*)malloc(sizeof(int)\* comm\_sz);

my\_id = 1;

start = MPI\_Wtime(); //calculate time from this point

//Loop to send values to each process - this is the hard part

for (k=count; k< number; k+=count){

MPI\_Send(array+k,count, MPI\_INT, my\_id, 0, MPI\_COMM\_WORLD);

//increment my\_id

my\_id++;

}

// Calculate sum for the values assigned to master process from the array

for(m=0;m<count;m++){

master\_sum=master\_sum+array[m];

}

// store sum of master process in correct index of summation array

summation[0]=master\_sum;

// Loop to receive sums from each slave process

for (j=1;j<comm\_sz;j++){

MPI\_Recv(&recv\_sum, 1, MPI\_INT, j, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

//store in correct index of summation array

summation[j]=recv\_sum;

}

//Call total\_sum to calculate sum of partial sums in summation

final\_sum=total\_sum(summation,comm\_sz);

finish = MPI\_Wtime(); // stop time at this point

printf("The summation for all numbers is %llu when N is equal to %lld \n", final\_sum, number);

printf("Total time taken is %f", finish - start);

}

//slave processes receive values sent by master process

else {

//declare recv\_data array to store "count" ints sent by master process to this process

int recv\_data[count];

//Receive the data from the master process

MPI\_Recv(recv\_data,count,MPI\_INT,0,0,MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);

//Call total\_sum to calculate the partial sum

for(p=0;p<count;p++)

{

partial\_sum =partial\_sum + \*(recv\_data+p);

}

//Send the sum of each slave process to master process

MPI\_Send(&partial\_sum ,1,MPI\_INT,0,0,MPI\_COMM\_WORLD);

}

MPI\_Finalize(); //MPI final

}

