**ITS 470**

**Fall 2017 Take-home test**

**Due: 10:59PM, December 10, 2017**

**Geyang Xie**

**30 points**

Pseudo code:

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

//init target number and assign the value;

int target **=** 1**;**

sscanf**(**argv**[**1**],** "%d"**,** **&**target**);**

//init the size of data array;

int n **=** 1000**;**

//init array 'nums 'size of n

array nums**[**n**];**

//read the file and put the data to nums

Read file**(**input1**.**dat**,** **&**nums**);**

//init MPI

Init\_mpi**(&**argc**,&**argv**);**

Init\_mpi\_size**(**MPI\_COMM\_WORLD**,&**nproc**);**

Init\_mpi\_rank**(**MPI\_COMM\_WORLD**,&**myrank**);**

Init\_MPI\_Bcast**(&**n**,** 1**,** MPI\_INT**,** 0**,** MPI\_COMM\_WORLD**);**

**if(**myrank **!=** 0**){**

//get how many tasks each processors does;

tasks **=** n **/** **(**nproc**-**1**);**

//calculate where to start and stop at

startAt **=** **((**myrank**-**1**)** **\*** tasks**)** **+** 1**;**

stopAt **=** myrank **\*** tasks**;**

int count **=** 0**;**

// count the total frequency for each processors

foreach**(** int i **:** from startAt to stopAt**){**

**if(**nums**[**i**]** **==** target\_number**){**

count**++;**

**}**

**}**

//reduce the count number to total count number

MPI\_Reduce**(&**count**,** **&**realTotal**,** 1**,** MPI\_DOUBLE**,** MPI\_SUM**,** 0**,** MPI\_COMM\_WORLD**);**

//print out result

**if(**myrank **==** 0**){**

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

printf**(**"The total frequency of %d is %d\n"**,**target**,** realTotal**);**

**}**

**}**

//close MPI

MPI\_Finalize**();**

**return** 0**;**

**}**

Source code :

#include "mpi.h"

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#define \_CRT\_SECURE\_NO\_DEPRECATE

#pragma warning (disable : 4996)

#define nums\_LIST\_MAX 1000

#define nums\_FILE "input1.dat"

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

int target **=** 1**;**

sscanf**(**argv**[**1**],** "%d"**,** **&**target**);**

int i**,** n**,** myrank**,** nproc**,** done **=** 0**;**

double h**,** x**;**

int sum**,** total**,** realTotal**;**

int tasks **=** 0**;**

int startAt **=** 0**;**

int stopAt **=** 0**;**

/\*

\* Read the .dat file to the array nums.

\*/

int nums**[**nums\_LIST\_MAX**];**

int count **=** 0**;**

int j **=** 0**;**

FILE **\***file**;**

file **=** fopen**(**nums\_FILE**,** "r"**);**

**if** **(!**file**)**

**{**

perror**(**"Error opening file"**);**

**return** **-**1**;**

**}**

memset**(**nums**,** 0**,** **sizeof(**nums**));**

**while** **(!**feof**(**file**)** /\* Check for the end of file\*/

**&&** **(**count **<** nums\_LIST\_MAX**))** /\* To avoid memory corruption \*/

**{**

fscanf**(**file**,** "%d"**,** **&(**nums**[**count**++]));**

**}**

fclose**(**file**);**

/\*

\* MPI caculation

\*/

MPI\_Init**(&**argc**,&**argv**);**

MPI\_Comm\_size**(**MPI\_COMM\_WORLD**,&**nproc**);**

MPI\_Comm\_rank**(**MPI\_COMM\_WORLD**,&**myrank**);**

n **=** 1000**;**

MPI\_Bcast**(&**n**,** 1**,** MPI\_INT**,** 0**,** MPI\_COMM\_WORLD**);**

**if** **(**n **==** 0**)**

done **=** 1**;**

**else**

**{**

**if(**myrank **==** 0**)**

**{**

printf**(**"number to find is %d\n"**,** target**);**

**}**

**else**

**{**

//get how many tasks each processors does

tasks **=** n **/** **(**nproc**-**1**);**

//calculate where to start and stop at

startAt **=** **((**myrank**-**1**)** **\*** tasks**)** **+** 1**;**

stopAt **=** myrank **\*** tasks**;**

sum **=** 0**;**

// The processor is doing jobs from startAt to stopAt index and count the total frequency for each processors

**for** **(**i **=** startAt**;** i **<=** stopAt**;** i**++)**

**{**

**if(**nums**[**i**]** **==** target**)**

**{**

sum**++;**

**}**

**}**

printf**(**"My id is %d and the frequency of %d is %d\n"**,** myrank**,** target**,** sum**);**

**}**

//reduce the count number to total count number

MPI\_Reduce**(&**sum**,** **&**realTotal**,** 1**,** MPI\_DOUBLE**,** MPI\_SUM**,** 0**,** MPI\_COMM\_WORLD**);**

//print out result

**if(**myrank **==** 0**){**

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

printf**(**"The total frequency of %d is %d\n"**,**target**,** realTotal**);**

**}**

**}**

MPI\_Finalize**();**

**return** 0**;**

**}**

