IN3200/IN4200 Home Exam 2, Spring 2020

This mandatory project (home exam No. 2) serves as a hands-on exercise of MPI programming. The requirements have been down-scaled (in comparison with the last year) due to the ongoing coronavirus pandemic.

1 The project: finding "triple-friends of 10"

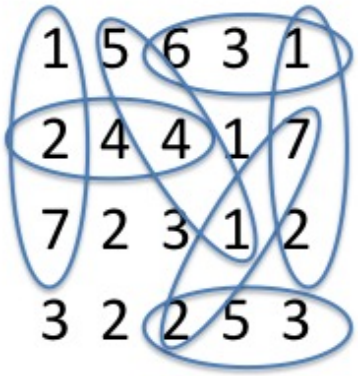

Figure 1: Examples of "triple-friends of 10" in a 2D table.

Given a 2D table that has M rows and N columns of non-negative integers, we want to find the total number of "triple-friends of 10". Here, a "triple-friend of 10" is defined as three neighboring numbers (in the horizontal, vertical, or diagonal directions) that sum up to 10. For example, there are 7 triple-friends in Figure 1.

Task 1

Write a serial C function

```
int count_friends_of_ten (int M, int N, int** v)
```

that returns the total number of "triple-friends of 10" inside the input 2D array `v`, which is of dimension $M \times N$.

This function should be stored in a file named `count_friends_of_ten.c`.

Task 2

Write a simple serial program that uses the above `count_friends_of_ten` function. More specifically, the serial program should allocate a 2D array of dimension $M \times N$, assign the array with appropriate integer values, and then call the `count_friends_of_ten` function to count the number of "triple-friends of 10" in the 2D array.

The serial program should be named `serial_main.c`.

Task 3

Write an MPI-parallelized C function

```
int MPI_count_friends_of_ten (int M, int N, int** v)
```

This function is to be called by **all** the MPI processes that are started by an MPI program (see the next task). However, **only** the master process (with rank 0) has the correct input argument values of `M`, `N`, and the 2D array `v` (of dimension $M \times N$). All the other MPI processes have zero values for variables `M` and `N`, and a `NULL` pointer `v` as the input arguments.

Inside `MPI_count_friends_of_ten`, the master process distributes array `v` appropriately, such that all the processes can join force and carry out the counting work in parallel. Finally, all the processes should return the total number of 'triple-friends of 10" in the 2D array `v`.

The function should be stored in a file named `mpi_count_friends_of_ten.c`.

Task 4

Write a simple MPI program that uses the above `MPI_count_friends_of_ten` function. A sketch of the MPI program is as follows:

```
int main (int argc, char **argv)
{
  int M=0, N=0, rank, num_triple_friends;
  int **v=NULL;

  MPI_Init (&argc, &argv);
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);

  if (rank==0) {
    // decide the values for M and N
    // allocate 2D array v and assign it with suitable values
  }

  num_triple_friends = MPI_count_friends_of_ten (M, N, v);

  printf("MPI rank <%d>: number of triple friends=%d\n",
         rank, num_triple_friends);

  if (rank==0)
    // deallocation of 2D array v

  MPI_Finalize ();
  return 0;
}
```

The MPI program should be named `mpi_main.c`.

2 Submission

Each student should submit a single tarball (`.tar`) or a single zip file (`.zip`). Upon unpacking/unzipping it should produce a folder named `IN3200_HE2_xxx` or `IN4200_HE2_xxx`, where `xxx` should be the candidate number of the student (can be found in StudentWeb). Inside the folder, there should be four `*.c` files described above. In addition, there should a `README.txt` (or `README.md`) file explaining how the compilation should be done, with additional comments if relevant. An accompanying `Makefile` is preferrable (but not mandatory).

The submission is through the Inspera system. Please see the course's semester webpage for info about the deadline. **No report or time measurements are needed in the submission.**

In case you don't have access to a computer that has MPI installed, please use a standard Linux server at Ifi (such as `login.ifi.uio.no`). The MPI-capable compiler there is `/usr/lib64/openmpi/bin/mpicc`, and the command for executing a compiled MPI program is `/usr/lib64/openmpi/bin/mpirun`. **Please only run short tests on any of Ifi's Linux servers.**

3 Grading

The grade of the submission will constitute 20% of the final grade of IN3200/IN4200. Grading of the submission will be based on the correctness, conformability (of filenames and function syntax), readability and speed of the implementations.