Homework Assignment 6 – due on Saturday, November 9 (Midnight)

Description of Assignment:

Complete an MPI program(matadd.c) that

(i) decomposes A and B on p_0 into local_A and local_B on all processors

compute <- (ii) computes local_C= local_A+local_B

(iii) composes local_C on all processors into C on p_0.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "mpi.h"

#define M 12
#define N 10

float **malloc_2d();
void grid();

main(int argc, char* argv[])
{
    float   A[M][N],   B[M][N],   C[M][N],   **local_A,
**local_B, **local_C;
    int np, inp, jnp, pid, local_M, local_N, *displs, *counts,
i, j, n, tag = 0;;
    MPI_Status status;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &np);
    MPI_Comm_rank(MPI_COMM_WORLD, &pid);

    grid(M, N, np, pid, &inp, &jnp);
    if (pid == 0) printf("%dx%d processors are used\n", inp,
jnp);

    local_M = M/inp;
    local_N = N/jnp;

    // initializaton of A and B
    if (pid == 0) {
        for (i=0; i<M; i++)
            for (j=0; j<N; j++) {
                A[i][j] = i*N;
                B[i][j] = j;
            }
    }
```

```
    local_A = malloc_2d(local_M, local_N);
    local_B = malloc_2d(local_M, local_N);
    local_C = malloc_2d(local_M, local_N);

    // (i) decompose A and B into local_A and local_B
    displs = (int*)malloc(sizeof(int)*np);
    counts = (int*)malloc(sizeof(int)*np);
    …

    // (ii) local_C = local_A + local_B
    …

    // (iii) compose local_C to C
    …

    // check the results
    if (pid == 0)
        for (i=0; i<M; i++) {
            for (j=0; j<N; j++)
                printf("%3.0f ", C[i][j]);
            printf("\n");
        }

    free(local_A);
    free(local_B);
    free(local_C);
    free(displs);
    free(counts);

    MPI_Finalize();
}
```

How to proceed:

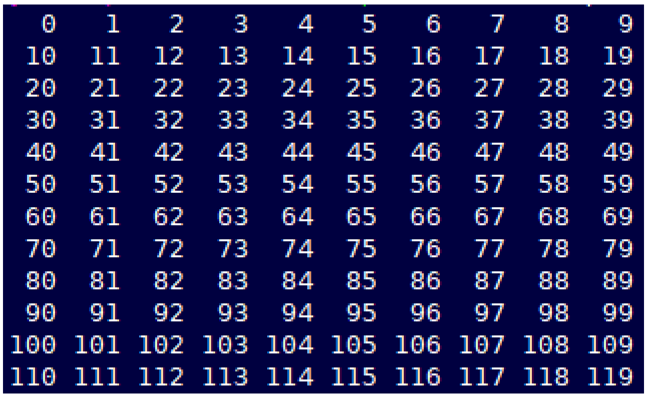
(i) Use MPI_Scatterv() and MPI_Gatherv() for decomposition and composition, respectively.

(ii) Copy /home/course/lib_2d.c into your working directory.

(iii) Compile “mpicc -o matadd matadd.c lib_2d.c”.

(iv) If numbers of processors are not divisible to M or N, a run-time error will be occurred

The program always prints following results regardless the number of processes.



```
  0   1   2   3   4   5   6   7   8   9
 10  11  12  13  14  15  16  17  18  19
 20  21  22  23  24  25  26  27  28  29
 30  31  32  33  34  35  36  37  38  39
 40  41  42  43  44  45  46  47  48  49
 50  51  52  53  54  55  56  57  58  59
 60  61  62  63  64  65  66  67  68  69
 70  71  72  73  74  75  76  77  78  79
 80  81  82  83  84  85  86  87  88  89
 90  91  92  93  94  95  96  97  98  99
100 101 102 103 104 105 106 107 108 109
110 111 112 113 114 115 116 117 118 119
```

Turnin the assignment:

After done your assignment, type **turnin** in your current working directory. You can retype the command at any time before the due date.