Homework Assignment 8 – due on Saturday, November 23 (Midnight)

Description of Assignment:

Write an MPI program(bound.c) that builds boundary data on the *local_A* matrixes in parallel and composes *local_A* to a matrix *A* on P_0.

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

#define M 12
#define N 10

int **malloc_2d();
void grid();

int main(int argc, char* argv[])
{                루프를 사용하지 않고 arry 초기화 하는 방법이다.
    int A[M][N] = {[0 ... M-1][0 ... N-1] = 0}, **local_A;
    int B[M][N] = {[0 ... M-1][0 ... N-1] = 0};
    int np, inp, jnp, pid, i, j, tag = 0;
    int dim_sizes[2], coord[2], wrap_around[2], reorder =
0;
    int local_M, local_N, x, y;
    MPI_Comm grid_comm;
    MPI_Datatype vector_t;
    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);

    // compute B on p0
    if (pid == 0) {
        for (j=0; j<N; j++) {
            B[0][j] = j;            // top
            B[M-1][j] = M-1+j; // bottom
        }
        for (i=0; i<M; i++) {
            B[i][0] = i;            // leftmost
            B[i][N-1] = N-1+i; // rightmost
        }

        printf("1 processor is used\n");
        for (i=0; i<M; i++) {
            for (j=0; j<N; j++)
                printf("%4d ", B[i][j]);
            printf("\n");
        }
    }
```

```
    dim_sizes[0] = ...;
    dim_sizes[1] = ...;
    wrap_around[0] = 0;
    wrap_around[1] = 0;

    MPI_Cart_create(...);
    MPI_Cart_coords(...);

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

    MPI_Type_vector(..., &vector_t);
    MPI_Type_commit(&vector_t);

    local_A = malloc_2d(local_M, local_N);

    for (i=0; i<local_M; i++)
        for (j=0; j<local_N; j++)
            local_A[i][j] = 0;

    // build boundary data in parallel
                                             // top
          Use coord[2]                       // bottom
                                             // leftmost
                                             // rightmost

    // composition
    if (pid == 0) {
        // copy local_A to A
        for (i=0; i<local_M; i++)
            for (j=0; j<local_N; j++)
                A[i][j] = local_A[i][j];

        ...
    }
    else {
        ...
    }

    if (pid == 0) {
        printf("\n%dx%d processors are used\n", inp, jnp);
        for (i=0; i<M; i++) {
            for (j=0; j<N; j++)
                printf("%4d ", A[i][j]);
            printf("\n");
        }
    }

    free(local_A);
    MPI_Finalize();
    exit(0);
}
```

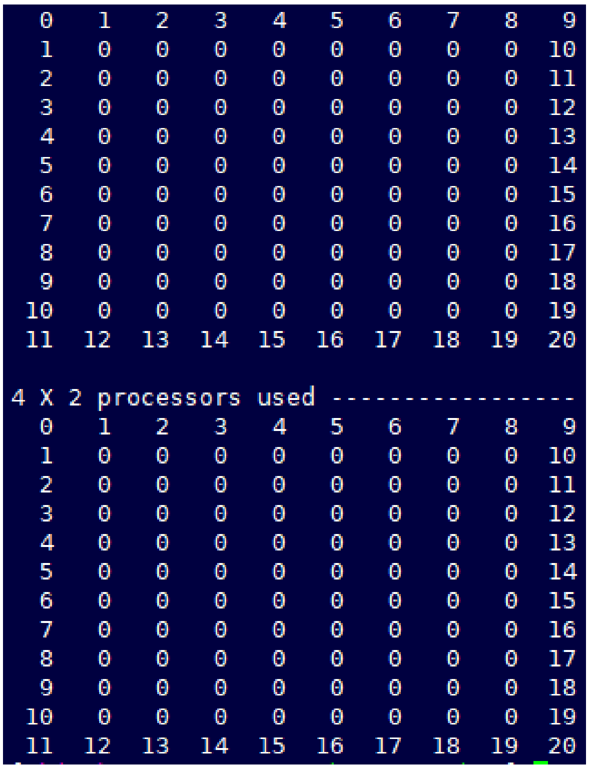
How to proceed:

(i) Complete the above program with functions marked with bold faced.

(ii) Copy /home/course/lib_2d.c to your working directory and compile with it.

```
mpicc -o bound bound.c lib_2d.c
```

(iii) With an appropriate number of processors are used, the program outputs the following result.



```
  0    1    2    3    4    5    6    7    8    9
  1    0    0    0    0    0    0    0    0   10
  2    0    0    0    0    0    0    0    0   11
  3    0    0    0    0    0    0    0    0   12
  4    0    0    0    0    0    0    0    0   13
  5    0    0    0    0    0    0    0    0   14
  6    0    0    0    0    0    0    0    0   15
  7    0    0    0    0    0    0    0    0   16
  8    0    0    0    0    0    0    0    0   17
  9    0    0    0    0    0    0    0    0   18
 10    0    0    0    0    0    0    0    0   19
 11   12   13   14   15   16   17   18   19   20

4 X 2 processors used ----------------
  0    1    2    3    4    5    6    7    8    9
  1    0    0    0    0    0    0    0    0   10
  2    0    0    0    0    0    0    0    0   11
  3    0    0    0    0    0    0    0    0   12
  4    0    0    0    0    0    0    0    0   13
  5    0    0    0    0    0    0    0    0   14
  6    0    0    0    0    0    0    0    0   15
  7    0    0    0    0    0    0    0    0   16
  8    0    0    0    0    0    0    0    0   17
  9    0    0    0    0    0    0    0    0   18
 10    0    0    0    0    0    0    0    0   19
 11   12   13   14   15   16   17   18   19   20
```

Turnin the assignment:

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