Lab7 – Grouping data

ex.1 Structure type

struct.c: send the following data from p_0 to p_1.

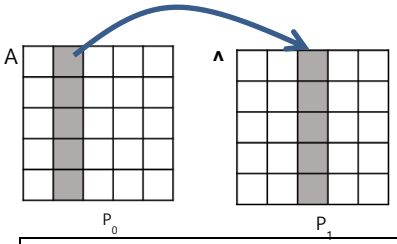
float A[2]; int i; char c;

A[0] = 11; A[1] = 22; I = 333; c = 'K';

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

int main(int argc, char* argv[])
{
    int pid, np, flag, tag = 0;
    MPI_Status status;

    MPI_Aint displacements[3];
    MPI_Aint start_address, address;
    MPI_Datatype typelist[3];
    int block_lengths[3];
    MPI_Datatype data_t;

    float A[2];
    int i;
    char c;

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

// COMPLETE THIS AREA

    MPI_Type_commit(&data_t);

    if (pid == 0) {
        A[0] = 11; A[1] = 22; i = 333; c = 'K';
        MPI_Send(A, 1, data_t, 1, tag, MPI_COMM_WORLD);
    }
    if (pid == 1) {
        MPI_Recv(A, 1, data_t, 0, tag, MPI_COMM_WORLD,
&status);
        printf("%2.1f %2.1f %d %c₩n", A[0], A[1], i, c);
    }
    MPI_Finalize();
}
```

ex.2 Vector type

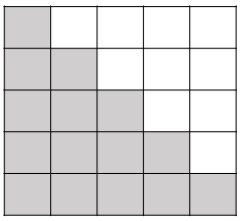
vector.c : send the column from p_0 to p_1. shown in the following figure.




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

int main(int argc, char* argv[])
{
    int pid, np, flag, tag = 0, i, j;
    MPI_Status status;
    MPI_Datatype column_mpi_t;
    float A[5][5];

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

     for (i=0; i<5; i++)
        for (j=0; j<5; j++)
            A[i][j] = 0.0;

// COMPLETE THIS AREA
    if (pid == 0) {
        for (i=0; i<5; i++)
                A[i][1] = 1.0;
        MPI_Send(&(A[0][1]), 1, column_mpi_t, 1, tag, MPI_COMM_WORLD);
    }
    if (pid == 1) {
        MPI_Recv(&(A[0][2]), 1, column_mpi_t, 0, tag, MPI_COMM_WORLD,
&status);
        for (i=0; i<5; i++) {
            for (j=0; j<5; j++)
                printf("%2.1f ", A[i][j]);
            printf("₩n");
        }
    }
    MPI_Finalize();
}
```

ex.3 Indexed type

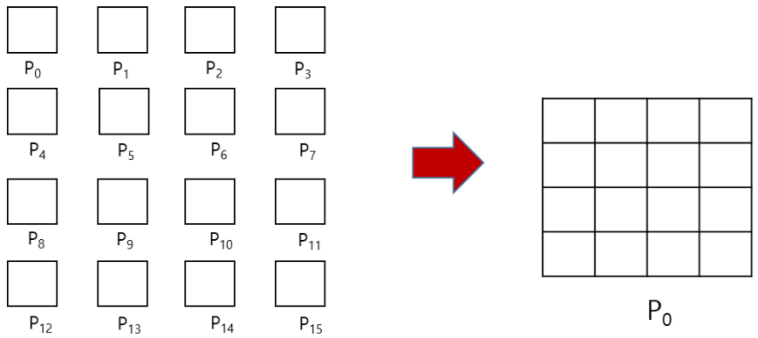
indexed.c: send the lower triangular of a matrix from p_0 to p_1. shown in the following figure.



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

#define N 5

int main(int argc, char* argv[])
{
    int pid, np, flag, tag = 0, i, j;
    MPI_Status status;
    MPI_Datatype index_t;
    int displacements[N];
    int block_lengths[N];

    float A[N][N], T[N][N];

    MPI_Init(&argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &pid);
    MPI_Comm_size(MPI_COMM_WORLD, &np);

    // COMPLETE THIS AREA
    if (pid == 0) {
        for (i=0; i<N; i++)
            for (j=0; j<N; j++)
                A[i][j] = i*N+j;
        MPI_Send(A, 1, index_t, 1, 0, MPI_COMM_WORLD);
    }
    if (pid ==1) {
        for (i=0; i<N; i++)
            for (j=0; j<N; j++)
                T[i][j] = -1.0;
        MPI_Recv(T, 1, index_t, 0, 0, MPI_COMM_WORLD, &status);
        for (i=0; i<N; i++) {
            for (j=0; j<N; j++)
                printf("%2.1f ", T[i][j]);
            printf("₩n");
        }
    }

    MPI_Finalize();
}
```

ex.4 Design the Lab#6 2d block composition program using MPI_Type_vector().

e.g. N=24, np2 = 16, np = sqrt(np2) = 4, local_N = N/np = 24/4 = 6



Submit compose.c when you are done.