CS 133 Homework 3
Dennis Zang, 704766877

1. Why do we need the concept of “communicator” in MPI? What is the default communicator? Assuming that we have 16 processors involved in the parallel computation, please provide the MPI code to create 4 communicators such that all processors with identical rank mod 4 are in the same communicator, where rank is the processor ID in the default communicator.

The concept of "communicator" is needed in the when different processors working in different virtual memory spaces need to communicate data to each other. The default communicator MPI_COMM_WORLD is the communicator that by default includes all processes running on the system.
Code:

```
MPI_Init(&argc, &argv);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int colColor = rank % 4;
MPI_Comm colComm;
MPI_Comm_split(MPI_COMM_WORLD, colColor, rank, &colComm);
MPI_Finalize();
```

2. Given a list L of k*N integers of value between 1 to m as the input evenly distributed among k processors stored in their local file systems, please write an efficient MPI program to generate the histogram h of list L at processor 0. Please make your function as efficient as possible, and highlight the MPI functions that you are using.

```
#include <cstdlib>
#include <ctime>
#include <iostream>
#include <mpi.h>

#define k 4
#define N 10000
#define m 100

void hist(const int L[k * N], int h[m])
{
	int worldRank, worldSize;
	MPI_Comm_rank(MPI_COMM_WORLD, &worldRank);
	MPI_Comm_size(MPI_COMM_WORLD, &worldSize);
	int partSize = (k * N) / worldSize;
	int partSizeRem = (((k * N) - 1) % partSize) + 1;
	int *LBuffer = new int[partSize];
	int *hBuffer = new int[m]();

	int *sendCount = new int[worldSize];
	int *displ = new int[worldSize];
	for(int i = 0, acc = 0; i < worldSize; i++)
	{
		if((i + 1) != worldSize)
			sendCount[i] = partSize;
		else
			sendCount[i] = partSizeRem;
		displ[i] = acc;
		acc += sendCount[i];
	}

	MPI_Scatterv(L, sendCount, displ, MPI_INT, LBuffer, sendCount[worldRank], MPI_INT, 0, MPI_COMM_WORLD);
	for(int i = 0; i < partSize; i++)
		hBuffer[LBuffer[i] - 1]++;
	MPI_Reduce(hBuffer, h, m, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

	delete[] sendCount;
	delete[] displ;
	delete[] LBuffer;
	delete[] hBuffer;
}

int main(int argc, char** argv)
{
	int rank;
	int *L;
	int *h;
	MPI_Init(&argc, &argv);
	MPI_Comm_rank(MPI_COMM_WORLD, &rank);
	if(rank == 0)
	{
		L = new int[k * N];
		srand(time(NULL));
		for(int i = 0; i < (k * N); i++)
			L[i] = (rand() % m) + 1;
		h = new int[m]();
	}
	MPI_Barrier(MPI_COMM_WORLD);
	hist(L, h);
	MPI_Barrier(MPI_COMM_WORLD);
	if(rank == 0)
	{
		for(int i = 0; i < m; i++)
			std::cout << (i + 1) << ":\t" << h[i] << std::endl;
		delete[] L;
		delete[] h;
	}
	MPI_Finalize();
}
```

3. For the E-cube algorithm for all-to-all personalized communication discussed in Lectures 7 and 8, if we implement it on a 4-dimension hypercube, how many steps will the algorithm goes through? At Step 7, which processor will processor 5 exchange messages?

For a 4-dimension hypercube, we have 16 nodes (0000-1111), so we will need a total of 15 steps to complete the all-to-all personalized communication (each node communicates with the 15 other nodes).
At step 7, processor 5 will communicate with (0101 ^ 0111) = 0010, or processor 2.

4. Consider the basic matrix multiplication algorithm for two NxN matrices A and B using KxK processors connected using a mesh network. Assume that each processor already has data the corresponding N/K x N/K sub-matrices of A and B, and only need to generate and store the resulting N/K x N/K sub-matrix locally. Please derive the isoefficiency relation and the scalability function. You may assume that N is a multiple of K.

sequential: O(N^2)
parallel computation: O(N^2 / K)
overhead: O(K + (N^2 / K))

isoefficiency: (p = K^2)
To(n,p) <= cT(n,1)
(p * T(n,p) - T(n,1)) <= cT(n,1)
pT(n,p) <= cT(n,1)
p(p + (n^2 / p)) <= cn^2
p^2 + n^2 <= cn^2
n >= p / c

scalability:
M(n) = n^2
M(p / c) / p = (p / (c^2))

5. In Lecture 9, we discussed the example shown on the right, which has a loop pipelining initiation interval equal to 2. If we only want to output d[SIZE], can you rewrite the code so that the II becomes 1?

```
i = 1;
prev = d[0]
for (i=1; i<=SIZE; i++) {
        prev = prev*v[i];
        d[i] = prev;
}
```