Matt Taylor
ECE 6105
Assignment 3

1.

Message Passing:

- Explicit communication via messages
- Loose coupling of program components

Data Parallel:

- Address space is treated globally
- Set of tasks operate on data, but independently on disjoint partitions

Shared Memory:

- Implicit communication via memory operations
- Tight coupling of program components

Distributed Shared Memory:

- Assumes global memory address space that is logically partitioned
- Portions of shared memory may have an affinity for a particular process

2.

'Affinity' component in *upc_forall* function is i/THREADS. Iteration i will be performed by thread [(i/THREADS) % THREADS].

i = 0, 1 , 2, 3 -> thread 0
i = 4, 5, 6, 7 -> thread 1
i = 8, 9, 10, 11 -> thread 2
i = 12, 13, 14, 15 -> thread 3
I = 16, 17, 18, 19 -> thread 0
And so on

X == N

Memory allocation:

Thread 0	**Thread 1**	**Thread 2**	**Thread 3**
image[0][0-63]	image[1][0-63]	image[2][0-63]	image[3][0-63]
image[4][0-63]	image[5][0-63]	image[6][0-63]	image[7][0-63]
…	…	…	…
image[60][0-63]	image[61][0-63]	image[62][0-63]	image[63][0-63]

Memory accesses:

	Thread 0	**Thread 1**	**Thread 2**	**Thread 3**
Shared	256	256	256	256
Remote	768	768	768	768

X == (N*N)/THREADS

Memory allocation:

Thread 0	**Thread 1**	**Thread 2**	**Thread 3**
image[0][0-63]	image[16][0-63]	image[0][0-63]	image[0][0-63]

image[1][0-63]	image[17][0-63]	image[1][0-63]	image[1][0-63]
…	…	…	…
image[15][0-63]	image[31][0-63]	image[47][0-63]	image[63][0-63]

Memory accesses:

	Thread 0	**Thread 1**	**Thread 2**	**Thread 3**
Shared	256	256	256	256
Remote	768	768	768	768

X == N/THREADS

Memory allocation:

Thread 0	**Thread 1**	**Thread 2**	**Thread 3**
image[0][0-15]	image[0][16-31]	image[0][32-47]	image[0][48-63]
image[1][0-15]	image[1][16-31]	image[1][32-47]	image[1][48-63]
…	…	…	…
image[63][0-15]	image[63][16-31]	image[63][32-47]	image[63][48-63]

Memory accesses:

	Thread 0	**Thread 1**	**Thread 2**	**Thread 3**
Shared	256	256	256	256
Remote	768	768	768	768

3.

```
#include <stdio.h>
#include <upc.h>
#define GNRL_SIZE 10//Assume the array size is 10
shared int arr0[GNRL_SIZE];   //shared array distributed acros threads
shared int partial_sums[THREADS];   //shared array with one integer per thread
shared int sum;                                 //sum will have affinity to thread 0
shared int mean;                        //mean will have affinity to thread 0
shared int var;                         //var will have affinity to thread 0

main(){
        int i, temp;
        upc_forall(i = 0; i < GNRL_SIZE; i ++; i){
                partial_sums[MYTHREAD] += arr0[i];          //calculate partial sums for each
thread
        }
        upc_barrier;    //wait until all partial sums calculated

        if(MYTHREAD == 0){
                for(i = 0; i < THREADS; i ++){
                        sum += partial_sums[i];                 //sum the partial sums
                }
                mean = sum/GNRL_SIZE;                           //calculate mean
                for(i = 0; i < GNRL_SIZE; i ++){
                        temp += (arr0[i]-mean)*(arr0[i]-mean);
                }
                var = temp/(GNRL_SIZE - 1);             //calculate variance
                printf("Mean = %i\tVariance = %i\n", mean, var);
        }
}
```

4.

```
#include <stdio.h>
#include <upc_relaxed.h>

#define N 512 //assume image is 512x512
shared [N*N/THREADS] unsigned char img [N][N] //Shared array with block size
N*N/THREADS
shared int histogram[256];
upc_lock_t*lock;

void initialize(void){
        lock = upc_all_lock_alloc();
        CHECK_MEM(lock);

}

int main(void){
        int i, j;
        initialize();
        upc_barrier;
        upc_forall(i = 0; i < N; i ++; i*THREADS/N){
                for(j = 0; j < N; j ++){
                        upc_lock(lock);
                        histogram[img[i][j]] ++;
                        upc_unlock(lock);
                }
        }
        upc_barrier;
        if(MYTHREAD == 0){
                //print histogram
        }
        return 0;
}
```