CS610: Programming for Performance
Assignment 2

Yash Gupta (190997)

1 Problem 1: Solution

The producers should only enqueue into Y if Y is not full. Similarly, consumers should only dequeue from Y if it is not empty. I have used 2 semaphores to wait and signal for the number of elements present in the queue and the number of empty spaces in the queue.

The consumers should terminate only after all the producers have terminated and the queue is empty. This is ensured by pushing a "poison pill", which is just an empty string, into Y after all the producers are finished. Each consumer terminates as soon as it sees the poison pill, without removing the poison pill from Y.

1.1 Compilation command

```
g++ 190997-prob1.cpp -pthread -lrt
```

2 Problem 2: Solution

I have used moodycamel ConcurrentQueue to implement Y. Semaphores are no longer needed. The poison pill is still needed for termination of consumers.

2.1 Compilation command

```
g++ -std=c++17 190997-prob2.cpp -pthread
```

3 Problem 3: Solution

3.1 Part (a)

To check if a loop permutation is valid, permute the dependence vectors according to the permutation. The permutation is valid if there are no “-” direction as the leftmost non–“0” direction in any direction vector, i.e., if any permuted vector is lexicographically negative, permutation is illegal.

The data dependences for the loop are given to be d1 = (1,0,-1,1), d2 = (1,-1,0,1), and d3 = (0,1,0,-1). This means that i, j and k cannot be the outermost loops as each of them have a corresponding -1 in d2, d1 and d3 respectively. This means that t will be the outermost loop in all valid permutations.

If t is the outermost loop, then d1 and d2 will always be valid as they have 1 in the leftmost position. So, we only need to consider the validity of d3. For d3 to be valid with t as the outermost loop, k will have to be inside i. Hence, the only 3 valid permutations (including the original permutation) are (t, i, j, k), (t, i, k, j) and (t, j, i, k).

3.2 Part (b)

Complete unroll/jam of a loop is equivalent to a loop permutation that moves that loop innermost, without changing order of other loops. If such a loop permutation is valid, unroll/jam of the loop is valid.

Unrolling and jamming k and j correspond to the permutations (t, i, j, k) and (t, i, k, j) respectively, both of which are valid as explained in part (a).

Unrolling and jamming i and t correspond to the permutations (t, j, k, i) and (i, j, k, t) respectively, both of which are invalid as explained in part (a).

Hence, only j and k are valid to unroll and jam.

3.3 Part (c)

A band of loops is fully permutable if all permutations of the loops in that band are legal. According to part (a), only 2D tiling of bands ij and jk are valid.

3.4 Part (d)

A transformation that reorders the iterations of a level-k loop, without making any other changes, is valid if the loop carries no dependence.

Loops t and i are not parallel as they have 1 in one or more of the dependence vectors. Only loop k is parallel as it does not carry the dependence in any of the dependence vectors.

3.5 Part (e)

The tikj form of the code is given below:

```c
int i, j, t, k;
for (t = 0; t < 1024; t++) {
    for (i = t; i < 1024; i++) {
        for (k = 1; k < i; k++) {
            for (j = t; j < i; j++) {
                S(t, i, j, k);
            }
        }
    }
}
```

4 **Problem 4:** Solution

4.1 Part (i)

I used the csews3 workstation. Its system description is:
L1 data cache size: 192 KiB
L1 instruction cache size: 192 KiB
L2 cache size: 1.5 MiB
L3 cache size: 12 MiB
CPU max frequency: 4600 MHz
CPU min frequency: 800 MHz