CS3211 – Parallel and Concurrent Programming

Practice Questions

Q1. T/F Questions

Are each of the following statements true or false? Justify your answer, and mention what programming language was assumed (if any).

a. The programming language memory model helps the programmer predict the behavior of the program independent of the architecture of the processor running on.
b. Deadlocks are always avoided when using synchronization primitives.
c. A concurrent program with livelocks will eventually complete when running on a single core processor (assume no SMT: the core does not support multiple hardware threads). Assume that the program creates, runs, and synchronizes multiple threads.
d. Starvation will not happen when atomic operations are used.
e. In Rust, a future can only be created using async functions or async blocks.

Q2. General Questions

Answer and explain the following questions:

a. In Go, implement a lock with channels and goroutines.
b. Explain one advantage for using scoped threads from Crossbeam in Rust.
c. Briefly explain one difference between variables passed to functions in Rust vs C++. How does this difference impact writing concurrent programs in these two languages?

Q3. C++ Memory Model

a. Consider 3 threads in C++ pseudo-code. There are 3 `std::atomic` variables, `x`, `y`, and `z` initialized to `0` before the threads have been created.

Thread 1	**Thread 2**

`x.store(1, memory_order_relaxed);`
`z.store(2, memory_order_release);`
`x.store(3, memory_order_release);`
	`while (y.load(memory_order_acquire)!=5);`
	`while (z.load(memory_order_relaxed)!=2);`
	`cout << x.load(memory_order_aquire);`

Thread 3	

`while (z.load(memory_order_acquire)!=2);`
`x.store(4, memory_order_relaxed);`
`y.store(5, memory_order_release);`	

Are the following statements TRUE or FALSE? Optionally, you may justify your answers.

Statement	True/False
i. The code will print 3, 5, or a value that is assigned to `x` after 1 or 5.	
ii. The code will never print 1.	
iii. The code will never print 3.	
iv. The code will never print 4.	
v. The code might run into an infinite loop.	
vi. If y would be a “normal” int, there wouldn’t be a data race for y.	

b. Consider the following C++ code snippet.

Line #	Code
1	`#include <atomic>`
2	`#include <thread>`
3	`#include <assert.h>`
4	`bool x=false;`
5	`std::atomic<bool> y;`
6	`std::atomic<int> z;`
7	`void write_x_then_y()`
8	`{`
9	`    x=true;`
10	`    std::atomic_thread_fence(std::memory_order_release);`
11	`    y.store(true,std::memory_order_relaxed);`
12	`}`
13	`void read_y_then_x()`
14	`{`
15	`    while(!y.load(std::memory_order_relaxed));`
16	`    std::atomic_thread_fence(std::memory_order_acquire);`
17	`    if(x)`
18	`        ++z;`
19	`}`
20	`int main()`
21	`{`
22	`    x=false;`
23	`    y=false;`
24	`    z=0;`
25	`    std::thread a(write_x_then_y);`
26	`    std::thread b(read_y_then_x);`
27	`    a.join();`
28	`    b.join();`
29	`    assert(z.load()!=0);`
30	`}`

Note: You can check the documentation for `std::atomic_thread_fence`. The actual exam would include documentation for `std::atomic_thread_fence`, if needed.

Are the following statements TRUE or FALSE? Optionally, you may justify your answers.

Statement		True/False
i.	The assert might fire.	
ii.	There is a data race for x. x must be made “atomic”.	
iii.	If y would be a “normal” bool, there wouldn’t be a data race for y.	
iv.	If z would be a “normal” int, there wouldn’t be a data race for z.	

Q4. Concurrent Programming Paradigms

In answering points a.-e. use the problem description of SortN**.**

SortN: Sorting an array of N integer numbers

We refer to this problem using the name **SortN**. Please find below an implementation of the sequential MergeSort algorithm that sorts an array `a` of size `n`.

Line#	Pseudo-code
1	`void merge(int a[], int size, int temp[]) {`
2	`  int i1 = 0;`
3	`  int i2 = n / 2;`
4	`  int it = 0;`
5	`  while(i1 < n/2 && i2 < n) {`
6	`  if (a[i1] <= a[i2]) {`
7	`  temp[it] = a[i1];`
8	`  i1 += 1;`
9	`  }`
10	`  else {`
11	`  temp[it] = a[i2];`
12	`  i2 += 1;`
13	`  }`
14	`  it += 1;`
15	`  }`
16	`  while (i1 < n/2) {`
17	`  temp[it] = a[i1];`
18	`  i1++;`
19	`  it++;`
20	`  }`
21	`  while (i2 < n) {`
22	`  temp[it] = a[i2];`
23	`  i2++;`
24	`  it++;`
25	`  }`
26	`  memcpy(a, temp, n*sizeof(int));`
27	`}`
28	
29	`void serial_mergesort(int a[], int n, int temp[]) {`
30	`  int i;`
31	`  if (n == 2) {`
32	`  if (a[0] <= a[1])`
33	`  return;`
34	`  else {`
35	`  SWAP(a[0], a[1]);`
36	`  return;`
37	`  }`
38	`  }`
39	`  serial_mergesort(a, n/2, temp);`
40	`  serial_mergesort(a + n/2, n - n/2, temp);`
41	`  merge(a, n, temp);`
42	`}`

Figure 1: Sequential Implementation of MergeSort

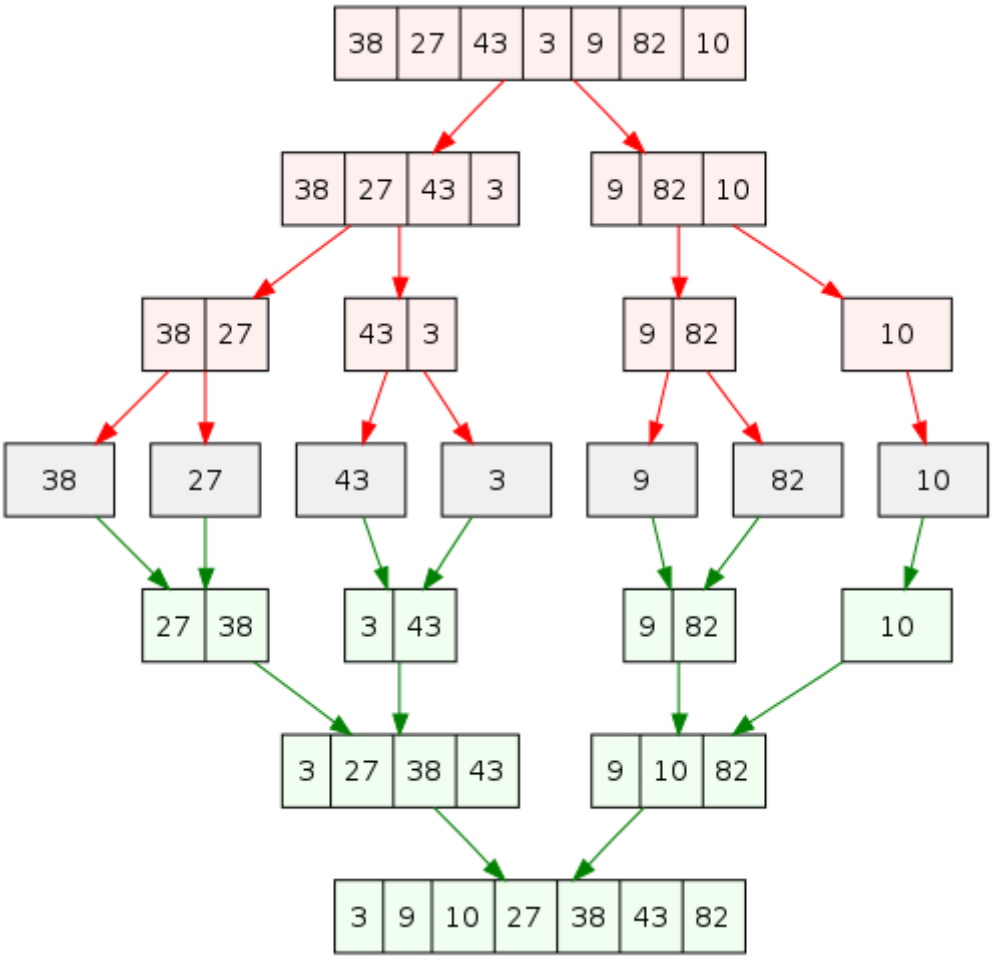

Figure 2: MergeSort Visual Representation

Suppose you are working on a concurrent implementation for SortN problem from Figure 1. Figure 2 shows a visual representation of the MergeSort algorithm.

a. Identify one part of SortN that could be implemented concurrently. Briefly explain the advantages and disadvantages of this change.

b. Sketch your concurrent implementation for SortN in Go. Your Go implementation may use any combination of the following goroutines, channels, shared data structures, and any utilities from `sync` package in Go (mutex, condition variable, pools, etc). Briefly describe the jobs (tasks) done concurrently by the goroutines and their interaction using channels. What would be the maximum level of concurrency that you expect in your implementation? (i.e. number of tasks that could potentially run in parallel)

c. Sketch your concurrent implementation for SortN in Rust using asynchronous programming paradigm. Briefly describe the jobs (tasks) done concurrently.

d. Sketch your concurrent implementation for SortN in modern C++, using threads and synchronization primitives. Focus on the part(s) of SortN that are implemented concurrently. You can use the line numbers to refer to the lines of pseudo-code. What would be the maximum level of concurrency that you expect in your implementation? (i.e. number of threads that could potentially run in parallel)

e. Compare your concurrent implementations from points b.,c.,d. Explain the advantages and disadvantages for each of them.