

**CS 4414 Final. 8 questions, 85 points total**  
**This exam will only be available from 9am on December 9 for 72 hours.**

Instructions: Type your answers directly on this Word document, then save the result as a PDF file and upload the PDF file to CMS. The exam is available for 72 hours but you are limited to 3 hours from when you download the exam until you upload the PDF file. No extra slack time is permitted unless you have an SDS letter on file with Professor Birman (if so, we follow whatever the SDS service asked us to do). Email the solution to Ken, Sagar or Alicia if you have an internet upload problem and are close to the 3 hour limit.

This is a **closed book, closed notes exam**. You are working at home but even so, we want you to approach the exam as if it was an in-person test with us watching and no use of devices permitted except for editing (there is one small exception: we don't mind if you decide to write and test the C++ code you use to answer the three coding questions). Do not use the Internet to research the questions, including from our web site or slides or Zoom lectures.

If you are the kind of person who often needs to ask questions during an exam, you may find it helpful to know that on Thursday from 2:30 to 5:30, Ken will be online ([here](#)) and can answer questions. Outside of that period post questions on Ed Discussions in private mode, but be aware that we can't monitor it continuously and we might not see your question in the 3-hour period while you need to finish the exam and upload a solution.

**Your name:** \_\_\_\_\_ **Your netid:** \_\_\_\_\_

1. [20 pts, 2 pts per T/F question] **Locking.** Suppose that you are developing a multithreaded C++ program that will run on a NUMA computer. Here is a list of "assertions". Mark the ones that are true by writing "true" next to them, and the ones that are false by writing "false". If a question says something that is partly true but partly false, you would write false.

	True/False	
a		A data object that is read and updated by two or more threads will need to be declared as <code>std::atomic</code> or protected by a <code>std::mutex</code> object.
b		It is <i>not</i> necessary to use a <code>std::mutex</code> to protect shared objects marked as "const".
c		When using the <code>std::scoped_lock</code> type to create a lock, you do need to specify a mutex object but do not need to give the <code>std::scoped_lock</code> a variable name, because you would never perform any operations on the object.
d		C++ monitors are created using a special template called <code>std::monitor</code> that you instantiate using lambdas for the reader and writer logic, and for deadlock detection.
e		With monitors, deadlock cannot occur.

f		If a program has lots of objects that need to be protected, you would need one mutex <i>per object</i> (for example if x and y are distinct objects, and both are accessed by a mix of reader and writer threads, you would need at least two mutex objects, one for x and one for y).
g		We only consider a process to be deadlocked if <i>all</i> the threads in it are involved in cyclic waits. If some threads are in a cyclic wait but others are running, this is a livelock.
h		In contrast to C++ programs, transactional systems that have deadlocks will often detect the cycle, abort the transaction that caused it (this will roll back the changes it made to variables while it was running), and then automatically restart that transaction.
i		The readers and writers pattern can be used to safely protect an STL object like a <code>std::list</code> or <code>std::map</code> that will be read by some threads and written by other threads.
j		With a C++ bounded buffer we can support a mix of producer and consumer threads. The solution also limits how far producers can get ahead of consumers.

2. [10 pts, 2pts per T/F question] **MapReduce Pattern.** These questions relate to the MapReduce pattern. Again, write true if the statement is correct and false if it is not correct.

	True/False	
a		MapReduce is valuable if a data set is so large that it can't fit on any one computer and must be split into pieces ("sharded") and spread over many computers.
b		Unlike parallel computing, a MapReduce computation is sequential. Every shard will be processed but the computations occur one by one, with each worker running in turn in an order decided by the leader.
c		In MapReduce, the term <i>shuffle</i> is used for the step in which each worker sends subresults to each of the other workers.
d		Before applying the <i>reduce</i> function, each worker sorts its received set of subresults, then groups them by key, and then the reduce function's role is to collapse each vector of values to some single result.
e		After MapReduce finishes, every worker has a complete and identical copy of the output of the entire computation

3. [10 pts, 2pts per T/F question] Debugging with `constexpr` and templates. True or false...

	True/False	
a		Template code is expanded (as much as possible) at compile time. As a result, gdb and the profiler won't necessarily be able to associate bugs that cause a crash to the proper line within the template, or give proper runtime cost-accounting for templated methods.
b		A constexpr expression cannot include variables that hold values the program reads from a user or from some other kind of input.
c		In gdb or gprof, a variable with a templated type will often have more type-signature content than you used to define that variable, because of expansion of default template type parameters and argument.
d		If a constexpr performs a zero divide, then when you run C++ to compile the code, the compiler will exit with a zero-divide exception.
e		gprof won't count the time C++ spends evaluating a constexpr when it prints a formatted profile report for the program.

4. [5 points] **DLLs.**

- a. [1 pt] Give a brief explanation in your own words of the “meaning” of the term DLL. We know that the letters stand for dynamically linked library. But what do DLLs do?

- b. [1 pt] Why is it important to compile a DLL with the “position-independent code” (PIC) g++ compiler flag?

- c. [3 pts] Suppose that A and B are two programs that share a DLL, and that it contains code but also some global variables. These objects are modified by the code as it runs. Do all the programs have private copies (so that A would see its own version, but not updates made by B), or shared copies (A sees B's updates)?

5. [10 points total] **Lambda Functions** For this problem, we will be using the following overload of `std::accumulate` defined in C++'s *numeric* library:

```
template< class InputIt, class T, class BinaryOperation >  
constexpr T accumulate( InputIt first, InputIt last, T init,  
                        BinaryOperation op );
```

This is the C++ equivalent of fold left that you might have seen in other contexts. Conceptually, it starts with an initial value of the “accumulator”, and successively updates it

by processing elements of a given sequence.

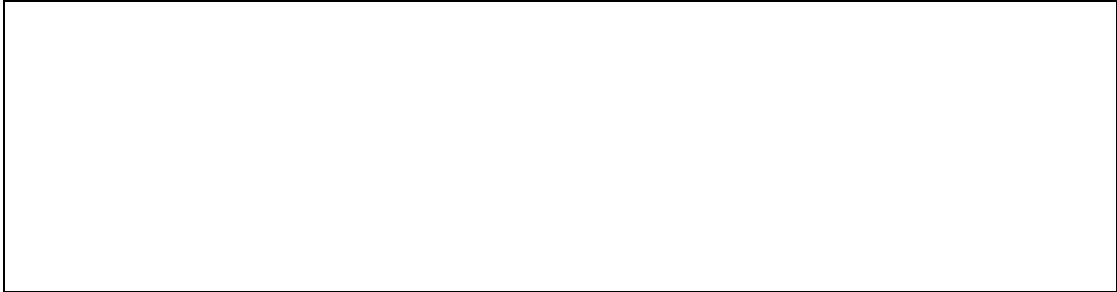
The first two arguments represent the range of the elements. For us, the range will be the entire sequence. For example, if our sequence is an `std::vector<int>` named `v`, we will pass in `v.begin()` and `v.end()`.

The third argument is an initial value of the accumulator and the fourth argument is a function that takes the current accumulator and an element of the sequence, and returns the new value of the accumulator. For instance, we can sum up all elements of an `std::vector<int>` by passing 0 as the initial value and a lambda function that takes two integers (the first integer is the accumulator type, the second is the element type) and returns their sum.

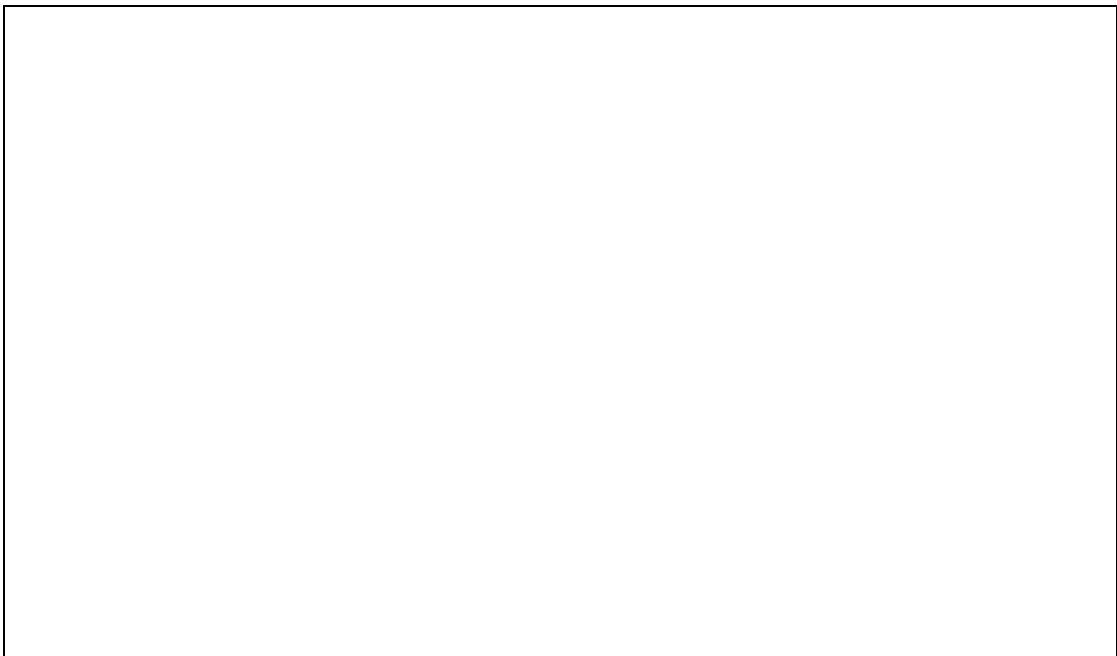
For each of the following tasks, write a one line call to `std::accumulate` to accomplish them. You are welcome to test your code in a C++ program to make sure that the syntax compiles and that your solution works. You can also make the code boxes bigger if your solution needs a little more space.

- a. **Modulo 7** Given an `std::vector<int>` object named `num`, where each element is a digit between 0 and 9 (inclusive), find the remainder mod 7 of the integer that has these same digits in sequence. For example, the number 635 can be represented as a vector of its digits 6, 3, and 5 (in that order). The result should be 5 (= 635 % 7).

- b. **Coverision to `std::string`** Given a `std::vector<char>` object named `str`, convert it to an object of `std::string`. For example, if the vector is [`'a'`, `'b'`, `'c'`] the result should be the string `"abc"`.



- c. **Distance from origin** Given a `std::vector<double>` named *point*, which represents a point in the  $n$ -dimensional space,  $\mathbb{R}^n$ , compute the square of its distance from the origin. For example, if the vector is `[1.6, 1.2]`, the result should be `4.0` ( $= 1.6^2 + 1.2^2$ ) (or an equivalent double value).



6. [10 points] Concurrency. Consider the following snippet of code:

```
std::atomic<int> counter(0);
std::function<void()> increment = [&counter]() {
    for(int i = 0; i < 100000; ++i) { counter = counter + 1; }
};
std::vector<std::thread> workers(4);
for(int i = 0; i < 4; ++i) {
    workers[i] = std::thread(increment);
```

```

}
for(int i = 0; i < 4; ++i) {
    workers[i].join();
}
std::cout << "counter = " << counter << std::endl;

```

As you can see, 4 threads concurrently execute the increment function in which they update an atomic int, *counter* 100K times each. Will the output of the program be 400K? Explain. If you think the answer is no, suggest a one line fix that does not involve using any other forms of locking or synchronization (that is, just use the given atomic integer). *Hint: Don't run this code as a way to try and find the answer. You could get very different results depending on your machine and how you compile it, and this could mislead you into giving a wrong answer. To answer this question you should read the code carefully, focusing on the increment method.*

7. [10 pts, 2 pts per question] C++ true/false questions

	True/False	
a		Execution of <code>ptr &amp;&amp; *ptr</code> will segfault for an integer pointer, <i>ptr</i> .
b		A functor in C++ is a class that defines the function call operator.
c		In order to use <code>std::map&lt;K,V&gt;::operator []</code> , for a user-defined class V, a default constructor for V must be implemented.
d		If a thread waits on a condition variable object, it may wake up even if it was not notified. For this reason it is important to recheck the wait condition.
e		Calling <code>notify_one</code> on a condition variable object will wake up the first thread that called wait on it (that is, <code>notify_one</code> guarantees a "FIFO" wake-up ordering).

8. [10 pts] C++ RAII (Resource Acquisition Is Initialization)

Consider a class *meeting* defined as followed:

```
class meeting {
// room where the meeting will be held std::shared_ptr<room> r;

public:
meeting(std::shared_ptr<room> r) : r(r) {}

void start() {
    r->turn_lights_on();
}

void end() {
    r->turn_lights_off();
}
};
```

Now suppose that you have been asked to modify this into an RAII implementation, in which allocating the object always turns the lights on, and they automatically go off when the object goes out of scope, Show us the revised code:



