CS4414 Prelim

October 14, 2021. 5 problems, 20 pts each

1. Consider a C++ program that reads a highly detailed engineering design for a bridge (in total, it loads 300 files containing about 3GB of data). Then the program carries out a structural analysis to evaluate stress on the various elements of the bridge, after which it creates a graphical visualization for the developer to study.

This program is currently much slower than it should be; you've done a pencil-and-paper estimate of how fast this program should run, and seems to be taking 10x longer than you would expect. Only the visualization runs at a reasonable speed. The computer used is a normal Intel-based desktop with lots of memory, but it lacks any kind of special accelerator hardware, except the built-in display graphics unit (which cannot be used as a GPU).

Upon investigation, you learn that the structural analysis turns employs a mechanical/physical simulation that is applied point by point at 5M distinct points on the bridge. At each point, a calculation is performed that sums up the stress contributions from each of 5000 "load sources" on the bridge (in total, 5M x 5000 stress contributions are evaluated).

a. [4 pts] In CS4414 lectures we discussed a number of possible causes for slow performance. List your "top four" ideas for things that could cause the structural stress analysis to run 10x too slowly.

For problems 1.b – 1.d, we want to focus on how you would identify the main cause.

b. [4 pts] How could the Linux "top" command help?
c. [4 pts] How could the Linux "time" command help?
d. [4 pts] How could the Linux "gprof" command help?

e. [4 pts] While investigating, you notice that the build command turns out to give g++ the argument –g, and not –O3. Explain what –g means, what –O3 means, and how this might contribute to the issue.

2. Still with reference to the bridge structural analysis from question 1, consider Gene Amdahl's famous formula: ***maximum speedup = 1/(1-p)***

a. [4 pts] Explain briefly what this formula means, in English (your wording should include a clear definition of the variable *p* that appears in the formula).
b. [4 pts] To apply Amdahl's law to our bridge analysis program, you would need to understand how to assign a value to *p*. How could *p* be estimated?
c. [4 pts] List two techniques that C++ programmers use to make a program more parallel.
d. [4 pts] Define the term SIMD. In what ways do modern computers support SIMD computing?
e. [4 pts] Suppose that after replacing –g with –O3 in the compilation line, the program is still not getting any benefit from SIMD instructions. What issues of *coding style* can prevent use of SIMD instructions?

3. File systems.

a. [5 pts] When we delete a Linux file, what is done to the file name, the inode, and the blocks containing the data?

b. [2 pts] We can access a disk in two ways: as a file system, or as a raw block storage device. Briefly define these two terms.

c. [5 pts] Describe a way to recover a deleted file by accessing a disk in raw "block" mode.

d. A *symbolic link* is a feature available from the Linux file system.

i. [2 pts] Briefly explain what a symbolic can be used to do.

ii. [2 pts] Suppose that A is a file, and that B is a symbolic link to that file. Someone edits the file using B to access it. Would a person who later looks at the file using A be able to see the edits? Explain.

iii. [2 pts] If A is deleted, could B still be used to access the file? Explain.

iv. [2 pts] If B is deleted, could A still be used to access the file? Explain.

4. C++ question: General understanding

a. [4 pts]

i. [2 pts] Why do we use namespaces in C++? Give just one main reason.

ii. [2 pts] What's the point of writing **#pragma once** at the top of the C++ header files?

b. [4 pts]

i. [2 pts] Write the main difference between static class variables and regular (non-static) class variables.

ii. [2 pts] Give one example scenario where you would use a static class variable.

c. [4 pts] This next question is based on a data structure you used in homework one, and that was discussed in recitation. Consider the following code:

```
#include <queue>

class Student {
    int score;

public:
    Student(int score) : score(score) {}
};

int main() {
    std::priority_queue<Student> students;
    Student x(97);
    students.push(x);
}
```

Now answer the following questions:

i. [2 pts] This will not compile. What will the error be?

ii. [2 pts] How will you fix this? Explain in words. No need to write code.

d. [4 pts] Assume that the following lines of code appear in the body of a method somewhere in your program:

```
int x = 3;
int& y = *new int(7);
int *p = &x;
int *q = &y;
```

Answer True or False for the following. No explanations required.

i. [1 pt] **&x** is a memory address on the stack

ii. [1 pt] **&y** is a memory address on the stack

iii. [1 pt] **&p** is a memory address on the stack

iv. [1 pt] **&q** is a memory address on the stack

e. [4 pts]

i. [2 pts] Define *memory leak* and explain very briefly how the C++ "new" operator can sometimes lead to memory leaks.

ii. [2 pts] C++ offers several protected wrappers for pointers that can avoid memory leaks. Two of those are **std::shared_ptr<T>** and **std::unique_ptr<T>.** How do they differ, and when might **std::shared_ptr<T>** be more appropriate?

5. C++ question: Exploring **std::vector<T>**

a. [4 pts]

i. [2 pts] Why does a vector sometimes need to call **malloc?**

ii. [2 pts] How does a vector avoid calling **malloc** every time a new element is inserted?

b. [4 pts] Consider the following two vectors:

```
std::vector<int> v1 = {1, 2, 3, 4};
std::vector<int> v2 = {7, 16};
```

Answer True or False for the following and explain your reasoning in one line.

i. [2 pts] **v1.size() == v2.size()**

ii. [2 pts] **sizeof(v1) == sizeof(v2)**

c. [4 pts] A vector provides O(1) random access. That is, given an arbitrary index **i** within the range of the vector, the element at that index can be retrieved in constant time. The vector achieves this by making sure that the elements are stored contiguously in memory and then finding the address of the element at index **i** by a formula that uses the address of the first element and the value of **i**.

Consider a vector of integer vectors i.e., **std::vector<std::vector<int>> vec;** with a fixed size of 5, and suppose that the internal vectors i.e., **vec[0], vec[1], ..., vec[4]** are automatically being resized because of insertion operations. Would this cause the outer vector to be resized? Explain.

d. [4 pts]

i. [2 pts] Explain the difference between **std::vector<T>::push_back(const T& value)** and **std::vector<T>::emplace_back(Args&&... args)**.

ii. [2 pts] The algorithm library in C++ (**#include <algorithm>**) provides many useful container functions. We will consider one such function, **std::sort(RandomIt first, RandomIt last, Compare comp)**, which sorts the elements in the range **first** to **last** using the given comparator function that compares two elements a and b. Our range will be the entire vector for this problem where the vector and its comparator function are defined as:

```
std::vector<int> vec = {60, 55, 39, 48, 50, 44, 57};
bool compare (int a, int b) {
     return std::abs(a - 50) < std::abs(b - 50);
}
```

Note: **std::abs (int n)** returns the absolute value of an integer. For example, **std::abs(-1) = 1**.

Write the vector contents after the call to sort:

```
std::sort(vec.begin(), vec.end(), compare);
```

e. [4 pts] The following program has a bug: the developer expected it to output 0 1 2 3 (each on its own line), but in fact it has no output at all! On checking, you find that nums.get_vec().size() is always 0 after calling push_back.

```
#include <iostream>
#include <vector>

class numbers {
    std::vector<int> num;

public:
    std::vector<int> get_vec() {
        return num;
    }
};

int main() {
    numbers nums;
    for(int i = 0; i < 4; ++i) {
        nums.get_vec().push_back(i);
    }
    for(int n : nums.get_vec()) {
        std::cout << n << std::endl;
    }
}
```

Explain the bug and describe a simple way to fix it (no need to show us the code). It is fine if your proposed fix requires some change to the given code.