Review Questions Answers

1) Justin is reconstructing his house and he has a piece of wood with some number of nails on it. A vector <int> w is representing the piece of wood and each element is a nail and the value of the element is the height of the nail. Justin has a broken hammer which can only pull up k nails to any height he wishes. His objective is to pull up k nails such that there are as many nails of the same height as possible. For example, w = {6, 9, 7, 9, 7, 9} and k = 2, he can pull any 2 of the {6, 7, 7} to height 9 so that there are 5 nails of height 9 in total.

Write a function to calculate the maximum number of nails of the same height after justin pull up k nails with the hammer. In the previous example, the function should return 5. Assume that k >= 0.

```
#include <algorithm>
#include <vector>

static uint SameHeight(const vector<uint> &w, uint k) {
  // Check if we need to do anything.
  if (w.size() <= k) {
    return w.size();
  }

  // Copy and sort the nail heights in ascending order
  vector<uint> copy(w);
  std::sort(copy.begin(), copy.end());

  // Iterate through the vector, keeping track of the most
  // frequent nail height.
  uint cur_count = 0, cur = copy[0], max = 0;
  for (uint i = 0; i < copy.size(); i++) {
    // Add to our current nail height if we
    // have a match
    if (cur == copy[i]) {
      cur_count++;
    } else {
      cur_count = 1;
      cur = copy[i];
    }

    // Update our maximum count
    if (cur_count + k > max) {
      max = cur_count;
      // Must check if our current count refers to the lowest
      // height nail. We can't pull anything lower than this nail
      // height so we cannot add anything to the frequency
      if (cur != copy[0]) {
        // Make sure that we are only adding what existed before us.
        uint remaining = i - cur_count + 1;
        max += (remaining >= k ? k : remaining);
      }
    }
  }

  // Since we only added what previously existed, we don't need to
  // do an additional check here.
  return max;
}
```

2) The age of Artificial Intelligence has come. There is an evil AI called SkyNut trying to conquer humanity by interrupting network communications. However, due to the lack of research fundings of the developers for SkuNut, it is not very smart. It does not have the ability to change the content of what people send, but it does know how to corrupt different network layers in any one computer or the internet system to interfere network communication. Your job is to identify which network layer it has corrupted in the following scenarios.

a) When the users enter any website link in the browser, it always redirects to www.skynutwins.com. Which layers (or layer) did Skynut interrupt? And more specifically, which part of that layer?
Application Layer, DNS

b) After part a is fixed, users can now send each other datas across the internet. However, once a while, even though the contents the users send are delivered to the correct destination on time, some of the contents would sometime be out of order or missing. Which layers (or layer) did Skynut interrupt? And more specifically, which part of that layer?
Transport layer, TCP. Keyword: out of order

c) Again, that previous part is now fixed by our genius UW CSE folks. We found another problem, however. We trace the packages that are sent to various destinations, all of which somehow would route to the Skynut's camping location, regardless of how far or close the destination is from the source. This could potentially be dangerous. Which layers (or layer) did Skynut interrupt?

Network layer, which determines where is the network to hop to.

d) Again, that previous parts are all fixed, and we realize there is one last problem. Sometimes the content of the files are messed up because the bits are randomly flipped. Which layers (or layer) did Skynut interrupt?
Physical layer for the bits corruption and linked lay for correcting the bit corruption.

3. It's the payday! It's time for UW to pay each of the 333 TAs their monthly salary. Each of the bank account is inside the *bank_accounts[]* array and the person who is in charged of paying the TAs is a 333 student and decided to use pthreads to pay the TAs by adding 1000 into each bank account. Here is the program the student wrote:

```
// Assume it includes all the necessary libraries and files
const int NUM_TAS = 10;

static int bank_accounts[NUM_TAS];
static pthread_mutex_t sum_lock;

void *thread_main(void *arg) {
  int *TA_index = reinterpret_cast<int*>(arg);

  pthread_mutex_lock(&sum_lock);
  bank_accounts[*TA_index] += 1000;
  pthread_mutex_unlock(&sum_lock);

  delete TA_index;
  return NULL;
}

int main(int argc, char** argv) {
  pthread_t thds[NUM_TAS];
  pthread_mutex_init(&sum_lock, NULL);

  for (int i = 0; i < NUM_TAS; i++) {
    int *num = new int(i);
    if (pthread_create(&thds[i], NULL, &thread_main, num) != 0) {
      /*report error*/
    }
  }
  pthread_mutex_destroy(&sum_lock);
  return 0;
}
```

a) Does the program increase the TAs' bank accounts correctly? Why or why not?

No its not correct. It needs to use pthread_join to wait for each thread to finish before exiting the main program. pthread_exit() might not be the best solution here. You want to check join output to make sure the transaction applied rather than exit and just trust the threads to finish successfully. Dolla dolla's important yo

b) Assumes that the all the problems, if any, are now fixed. The student discovers that the program she wrote is kinda slow even though its a multithreaded program. Why might it be the case? And how would you fix that?

Because there is a lock over the entire bank account array, so only one thread can increase the value of one account at a time and there its no difference from incrementing each account sequentially. To fix this, we can have one lock per account so that multiple threads can increment the account at the same time.

4.

a) List some reasons why it's better to use multiple threads within the same process over multiple processes of the same program

Processes are more expensive since they need their own address space.
Threads are more lightweight.
Switching processes are expensive because of context switch. Switching threads is cheap

b) Which registers will for sure be different between 2 threads that are executing different functions??

Stack pointer and program counter. (Each thread has its own stack) (Different functions so different program counters)

c) How does the OS distinguish the threads?

Thread IDs