CISS 370: Operating Systems
Assignment 2
Due date: February 10, 2021

For some of the following problems, you will need to download the thread library from *https://homes.cs.washington.edu/~tom/code/*. The comment at the top of *threadHello.c* explains how to compile and run a program that uses this library.

```
#include <stdio.h>
#include "thread.h"

static void go(int n);

#define NTHREADS 10
static thread_t threads[NTHREADS];

int main(int argc, char **argv) {
    int i;
    long exitValue;

    for (i = 0; i < NTHREADS; i++){
        thread_create(&(threads[i]), &go, i);
    }
    for (i = 0; i < NTHREADS; i++){
        exitValue = thread_join(threads[i]);
        printf("Thread %d returned with %ld\n",
                        i, exitValue);
    }
    printf("Main thread done.\n");
    return 0;
}

void go(int n) {
    printf("Hello from thread %d\n", n);
    thread_exit(100 + n);
    // Not reached
}

% ./threadHello
Hello from thread 0
Hello from thread 1
Thread 0 returned 100
Hello from thread 3
Hello from thread 4
Thread 1 returned 101
Hello from thread 5
Hello from thread 2
Hello from thread 6
Hello from thread 8
Hello from thread 7
Hello from thread 9
Thread 2 returned 102
Thread 3 returned 103
Thread 4 returned 104
Thread 5 returned 105
Thread 6 returned 106
Thread 7 returned 107
Thread 8 returned 108
Thread 9 returned 109
Main thread done.
```

Figure 4.6: Example multi-threaded program using the simple threads API that prints "Hello" ten times. Also shown is the output of one possible run of this program.

Question 1 (20 points)

Download *threadHello.c*, compile it, and run it several times. What happens when you run it? Do you get the same result if you run it multiple times? What if you are also running some other demanding processes (e.g., compiling a big program, playing a Flash game on a website, or watching streaming video) when you run this program?

Question 2 (20 points)

For the *threadHello* program in Figure 4.6, suppose that we delete the second for loop so that the main routine simply creates NTHREADS threads and then prints “Main thread done.” What are the possible outputs of the program now. Hint: Fewer than NTHREADS+1 lines may be printed in some runs. Why?

Question 3 (20 points)

Write a program that has two threads. Make the first thread a simple loop that continuously increments a counter and prints a period (“.”) whenever the value of that counter is divisible by 10,000,000. Make the second thread repeatedly wait for the user to input a line of text and then print “Thank you for your input.” On your system, does the first thread makes rapid progress? Does the second thread respond quickly?

Question 4 (20 points)

For the *threadHello* program in Figure 4.6, the procedure go() has the parameter *np* and the local variable *n*. Are these variables per-thread or shared state? Where does the compiler store these variables’ states?

Question 5 (20 points)

For the *threadHello* program in Figure 4.6, the procedure main() has local variables such as *i* and *exitValue*. Are these variables per-thread or shared state? Where does the compiler store these variables?