CSCD 340
Lab 8

In class we discussed pthread code to print hello world to the screen and to sum some values.
NOTE: all output is saved in a single PDF. You must number the outputs so it matches to the problem it is addressing.

PTHREAD FUNCTIONS

First, you will need to be able to create new threads, and you can do just that with the following PThread API routine:

- int pthread_create(pthread_t *thread, pthread_attr_t attr, void* (*start_routine)(void*), void *arg)

The first argument, pthread_t *thread, allows the calling thread to keep a structure that contains data relevant to the thread it creates; you can think of it as keeping a reference. We will not use the pthread_attr_t attr argument. The start_routine argument is a function pointer to the function that the thread will start in. In C you will always write a function name for this argument. Finally, the arg argument is a value of ambiguous type (thus the void * typing) that will be passed to the start_routine that is called when the thread begins.

A typical call to create a thread looks like:

```
pthread_t  the_other_thread;`
pthread_create(&the_other_thread, NULL, startingFunction, NULL);
```

The call above creates and runs a new independent thread of execution which starts at the top of startingFunction.

Other pthread functions you might need are:

- int pthread_join(pthread_t thread, void **value_ptr);
- void pthread_exit(void *value_ptr);

pthread_join - When thread A joins thread B, thread A will not continue until thread B has completed and exited.

The *pthread_exit* function terminates the calling thread and makes the value value_ptr available to any successful join with the terminating thread.

TO COMPLETE

1) I have provided code for the producer. Your task is:
    - Write the consumer code and modify the producer code by adding printf statements to illustrate what is going on. For example: Producer creating widget 1234, and placing it in the buffer or buffer is empty consumer is blocking.
    - The producer code is:

```
for(x = 1; x <= MAX; x++)
{
        pthread_mutex_lock(&the_mutex);
        while(buffer != 0)
        {
                pthread_cond_wait(&condp, &the_mutex);
        }// end while

        buffer=x;
        pthread_cond_signal(&condc);
        pthread_mutex_unlock(&the_mutex);
}// end for
```

    - Name the file that contains your basic Producer/Consumer code cscd340Lab8prob1.c
    - Compile and run the code and save the output in a file named cscd340Lab8out.pdf. NOTE: I don't need every line of output, just enough to see what is going on.
    - In the PDF complete the following:
        - In your own words explain what pthread_cond_wait does (I don't want a copy of the man page – hence in your own words)
        - Why is the first parameter condp instead of condc for the producer and condc instead of condp for the consumer?
        - Your output should be along the lines of produce then consume, produce then consume, produce then consume, etc – why does this behavior occur.

2) In previous problem, the producer produces and the consumer consumes for MAX (100) times. Modify the code so there is a true buffer that can potentially fill up.
    - Each for loop will be replaced by an infinite while loop
    - Buffer will need to be modified so it is an array of size MAX.
    - Name your file cscd340Lab8prob2.c
    - Compile and run your program and capture the output and save the output in the PDF.
    - Remember the program runs for ever so let it run for a few seconds and then kill it.

3) Using cscd340Lab8prob2.c as a starting point, add code to main to create 2 producers and 4 consumers.
    - Name this file cscd340Lab8prob3.c
    - Run the code and save the output in the PDF. Again just enough output to illustrate what is going on.
    - Add a section to the PDF that clearly explains the output and what is happening.

4) Using cscd340Lab8prob2.c as a starting point, add code to main to create 5 producers and 2 consumers.

- Name this file cscd340Lab8prob4.c
- Run the code and save the output in the PDF. Again just enough output to illustrate what is going on.
- Add a section to the PDF that clearly explains the output and what is happening, including a thoughtful explanation concerning the difference if there are more producers than consumers or more consumers than producers.

TO TURN IN

A zip

- All your C files
- A Makefile with targets for each individual problem (prob1, prob2, prob3, prob4)
- Your PDF

You will submit a zip file named your last name first letter of your first name lab8.zip
(Example steinerslab8.zip)