Homework 3

Tom Kelliher, CS 311

30 points, due Mar. 9, 2012

This problem is an extension of Exercise 4.17 in the textbook. Write a C program that uses the pthreads library to solve this problem. The program should take one command line argument that specifies the length of the Fibonacci sequence to generate. The argument should be between 2 and 99, and the parent thread should check the validity of this argument.

The parent thread will create *two* child threads to generate the elements of the Fibonacci sequence. One child will generate the even elements of the sequence and the other child will generate the odd elements of the sequence. Note that this makes each child dependent upon the other, as odd element fib_{2i+1} of the sequence cannot be generated until even element fib_{2i} of the sequence has been generated. A similar relationship holds for elements fib_{2i} and fib_{2i-1} of the sequence.

Once each child thread has generated its half of the Fibonacci sequence and exited, the parent thread should then print the generated sequence and exit.

Example

In addition to the pthreads library examples in the textbook, here is one more:

```
/******************************************************************
 * helloThreaded.c
 *
 * From https://computing.llnl.gov/tutorials/pthreads/.
 * Modified by Tom Kelliher
 *
 * This program demonstrates the pthreads library.  To compile under
 * Linux, a command similar to
 *
 *    gcc -o helloThreaded -lpthread helloThreaded.c
 *
 * should suffice.
 *
 * This program is a simple demonstration of the pthreads library.
 * The main thread creates NUM_THREADS child threads, each of which
 * prints a "Hello world" message.  By performing a join on each child
 * thread, the main thread hangs around until each child thread has
 * exited.
 ******************************************************************/
```

```
#include <stdlib.h>
#include <pthread.h>
#include <stdio.h>


#define NUM_THREADS 10


/* Function prototypes */

void *printHello(void *);


/*****************************************************************
 * main()
 *****************************************************************/

int main (int argc, char *argv[])
{
   pthread_t threads[NUM_THREADS];   /* Array to store thread IDs. */
   int i;
   int rc;   /* Return code from function call. */

   for (i = 0; i < NUM_THREADS; i++)
   {
      printf("In main: creating thread %d.\n", i);

      /* Create a thread.  The first argument is a pointer to a variable
       * of type pthread_t and is used to store the newly-created
       * thread's ID.  The second argument is used to specify attributes
       * of the new thread.  Passing a NULL pointer will result in the
       * use of default attributes.  The third argument is a pointer
       * to the function that the thread will initially start in.  This
       * function must take a single argument of type pointer to void and
       * must return a value of type pointer to void.  The final argument
       * is the value to pass to the initial function.
       *
       * In this specific instance, each thread is passed its index
       * within the threads array; this value is used to identify the
       * various threads.
       */

      rc = pthread_create(&threads[i], NULL, printHello, (void *) i);

      if (rc)
      {
         printf("ERROR; return code from pthread_create() is %d.\n", rc);
         exit(-1);
```

```
      }
   }

   /* Wait for each of the child threads to exit.  The join call returns
    * immediately if the child has already exited.  In this example, we
    * we are not using the return value mechanism available to us.
    */

   for (i = 0; i < NUM_THREADS; i++)
      pthread_join(threads[i], NULL);

   printf("Main thread exiting.\n");
   return 0;
}


/*****************************************************************
 * printHello()
 *****************************************************************/

void *printHello(void *threadid)
{
   int tid;   /* Thread ID */

   /* threadid was originally an int, cast to (void *).  Cast it back to
    * int.
    */

   tid = (int) threadid;
   printf("Hello World! It's me, thread #%d!\n", tid);
   pthread_exit(NULL);   /* Not returning a value. */
}
```

Project Turn-In

Email your source code file to `kelliher[at]goucher.edu` by the beginning of class on the 9th. A 15% penalty will be applied for each day the project is late. Saturday is not a day.