PROGRAMMING TOOLS LAB ASSIGNMENT
Week # 9

1. Write a program that creates 10 threads. Have each thread execute the same function and pass each thread a unique number. Each thread should print “Hello, World (thread n)” five times where ‘n’ is replaced by the thread’s number. Use an array of pthread_t objects to hold the various thread IDs. Be sure the program doesn’t terminate until all the threads are complete.

2. Write a program to create a thread with default attributes and then change the priority to HIGHPRIORITY.

3. Write a program for matrix multiplication using multiple threads, where each thread will perform multiplication of one row from first matrix A and one column from second matrix B. Therefore, each entry in the resultant matrix will be a result of a specific thread, as $[R]_{m*l} = [A]_{m*n} . [B]_{n*l}$. Hence the total numbers of threads are m*l. Dimensions of the matrix should be given as arguments to the main thread.

4. Write a program that computes the square roots of the integers from 0 to 99 in a separate thread and returns an array of doubles containing the results. In the meantime the main thread should display a short message to the user and then display the results of the computation when they are ready.

5. (a) Write a program to simulate deposit/withdraw activities on a banking account: Initialize the beginning balance (global variable) to 1 million, withdraw 600 thousands, and then deposit 500 thousands. Create two Posix threads in main(), which call the withdraw and the deposit functions respectively. Both withdraw and deposit functions have one parameter, which represent the amount to withdraw or deposit. You can create these two threads in any order to perform withdraw and deposit action. However, before you create the second thread, use pthread_join() to wait for the first thread to terminate. Finally print out the ending balance.

   (b) Move the calls to pthread_join() function after the creation of both pthreads. Run the program several times. Do you see different result?

   (c) Use pthread-mutex_lock() and pthread-mutex_unlock() functions to ensure mutual exclusion between the two pthreads. Check the ending balance now.

6. The Program should demonstrates the use of several Pthread condition variable (pthread_mutex_t, pthread_cond_t) routines. The main routine creates three threads. Two of the threads perform work and update a varible "race".The first thread add 3 in the “race” and second thread substract 1 from the “race”. The third thread waits until the count variable reaches 100.