**CSE381 Lab 5**

1. **Objective**

* Understand the synchronization problems
* Practice using lock to solve synchronization problems
* Practice using condition variable to solve synchronization problems

1. **Setup**

1)Download the file lab5.tar from Canvas.

2)Extract files from the archive lab5.tar using the following command:

tar xvf lab5.tar

1. **Synchronization of Child Thread Creations**

**(20 points)**

1. Copy the threadPrint.cc to threadPrintCV.cc using the following command:

cp threadPrint.cc threadPrintCV.cc

1. Please use the condition variable and modify threadPrintCV.cc to ensure that child threads start printing after all child threads are running.
2. Compile the threadPrintCV using the following command:

make threadPrintCV

1. Before using the condition variable, you need to include the header file CV.h, by adding the following statement to your file:

#include “CV.h”

1. Submit the threadPrintCV.cc

**4 Priority Lock (40 points)**

Read the FIFO blocking bounded queue example in the textbook (section 5.6.3) and finish the following problem.

“Before entering a priority critical section, a thread calls PriorityLock::enter(priority). When the thread exits the critical section, it calls PriorityLock::exit(). If several threads are waiting to enter a priority critical section, the one with the numerically highest priority should be the next one allowed in. Implement PriorityLock using locks and condition variables and following the programming standards defined in this chapter.”

1. Implement the Priority Lock with a header file and an implantation file

Must use PriorityLock.h and PriorityLock.cc for them.

Submit these two files.

(Hint 1: you may use a priority queue. Here are references for the priority queue:

1. http://www.cplusplus.com/reference/queue/priority\_queue/
2. http://web.eecs.utk.edu/~leparker/Courses/CS302-Fall06/Notes/PQueues/priority.cpp

)

(Hint 2:

In this excise, you need to precisely wake up the desired waiting thread. To do that, you will create a dedicated condition variable for each thread. In your enter(priority ), a condition variable is inserted to priority queue. In the STL priority queue, each entry has a key and value. In this example, the key is the priority and the value is the condition variable.

In the C++, you can insert a pair to the priority queue. A pair include both priority and CV in this problem. The following example shows how to use pair in priority queue.

#include <queue>

#include <string>

#include <iostream>

#include <utility>

int main()

{

//using pii = std::pair< int, std::string > ;

std::priority\_queue< std::pair< int, std::string > > pq ;

pq.push( std::make\_pair(45, "abc") ) ;

pq.push( std::make\_pair(78, "abc") ) ;

pq.push( std::make\_pair(12, "xyz") ) ;

pq.push( std::make\_pair(25, "pqr") ) ;

pq.push( std::make\_pair(45, "def") ) ;

while( !pq.empty() )

{

std::pair< int, std::string > pair = pq.top() ;

std::cout << pair.first << ',' << pair.second << '\n' ;

pq.pop() ;

}

}

)