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

Void* student_actions( void* student_id );
Void* ta_actions();

#define NUM_WAITING_CHAIRS 3
#define DEFAULT_NUM_STUDENTS 5

Sem_t sem_students;
Sem_t sem_ta;
Pthread_mutex_t mutex_thread;

Int waiting_room_chairs[3];
Int number_students_waiting = 0;
Int next_seating_position = 0;
Int next_teaching_position = 0;
Int ta_sleeping_flag = 0;

Int main( int argc, char **argv ){

        Int I;
        Int student_num;

        If (argc > 1 ) {
                If ( isNumber( argv[1] ) == 1) {
```

```
                Student_num = atoi( argv[1] );
        }
        Else {
                Printf("Invalid input. Quitting program.");
                Return 0;
        }
}
Else {
        Student_num = DEFAULT_NUM_STUDENTS;
}

Int student_ids[student_num];
Pthread_t students[student_num];
Pthread_t ta;

Sem_init( &sem_students, 0, 0 );
Sem_init( &sem_ta, 0, 1 );

//Create threads.
Pthread_mutex_init( &mutex_thread, NULL );
Pthread_create( &ta, NULL, ta_actions, NULL );
For( I = 0; I < student_num; i++ )
{
        Student_ids[i] = I + 1;
        Pthread_create( &students[i], NULL, student_actions, (void*) &student_ids[i] );
}

//Join threads
Pthread_join(ta, NULL);
```

```
	For( I =0; I < student_num; i++ )
	{
		Pthread_join( students[i],NULL );
	}

	Return 0;
}

Void* ta_actions() {

	Printf( “Checking for students.\n” );

	While( 1 ) {

		//if students are waiting
		If ( number_students_waiting > 0 ) {

			Ta_sleeping_flag = 0;
			Sem_wait( &sem_students );
			Pthread_mutex_lock( &mutex_thread );

			Int help_time = rand() % 5;

			//TA helping student.
			Printf( “Helping a student for %d seconds. Students waiting = %d.\n”, help_time,
(number_students_waiting – 1) );
			Printf( “Student %d receiving
help.\n”,waiting_room_chairs[next_teaching_position] );
```

```
                    Waiting_room_chairs[next_teaching_position]=0;
                    Number_students_waiting--;
                    Next_teaching_position = ( next_teaching_position + 1 ) %
NUM_WAITING_CHAIRS;

                    Sleep( help_time );

                    Pthread_mutex_unlock( &mutex_thread );
                    Sem_post( &sem_ta );

                }
                //if no students are waiting
                Else {

                    If ( ta_sleeping_flag == 0 ) {

                        Printf( “No students waiting. Sleeping.\n” );
                        Ta_sleeping_flag = 1;

                    }

                }

        }

}

Void* student_actions( void* student_id ) {
```

```
Int id_student = *(int*)student_id;

While( 1 ) {

        //if student is waiting, continue waiting
        If ( isWaiting( id_student ) == 1 ) { continue; }

        //student is programming.
        Int time = rand() % 5;
        Printf( "\tStudent %d is programming for %d seconds.\n", id_student, time );
        Sleep( time );

        Pthread_mutex_lock( &mutex_thread );

        If( number_students_waiting < NUM_WAITING_CHAIRS ) {

                Waiting_room_chairs[next_seating_position] = id_student;
                Number_students_waiting++;

                //student takes a seat in the hallway.
                Printf( "\t\tStudent %d takes a seat. Students waiting = %d.\n", id_student,
number_students_waiting );
                Next_seating_position = ( next_seating_position + 1 ) %
NUM_WAITING_CHAIRS;

                Pthread_mutex_unlock( &mutex_thread );

                //wake TA if sleeping
                Sem_post( &sem_students );
```

```
                    Sem_wait( &sem_ta );

                }
                Else {

                    Pthread_mutex_unlock( &mutex_thread );

                    //No chairs available. Student will try later.
                    Printf( "\t\t\tStudent %d will try later.\n",id_student );

                }

            }

}

Int isNumber(char number[])
{
   Int I;
                For ( I = 0 ; number[i] != 0; i++ )
   {
      If (!isdigit(number[i]))
         Return 0;
   }
   Return 1;
}

Int isWaiting( int student_id ) {
            Int I;
```

```
        For ( I = 0; I < 3; i++ ) {
                If ( waiting_room_chairs[i] == student_id ) { return 1; }
        }
        Return 0;
}
```