EECS 678 – Operating Systems – Fall 2020
Quiz – 5

Name: Prasad Kulkarni

Lab Time: (a) M 12:00pm, (b) M 4:00pm, (c) W 9:00am, (d) W 4:00pm, (e) F 12:00pm

1. **True or False (1 point):** A *synchronous* signal caused by a divide-by-zero error will only be delivered to the thread that caused the signal.

2. **True or False (1 point):** Using *OpenMP* requires the user to make changes to the source program code.

3. **True or False (1 point):** Disabling interrupts to enforce mutual exclusion locks will require superuser priviledges on uniprocessor systems.

4. **True or False (1 point):** *Any* solution to the critical section problem will only allow one process to execute in the critical section at a time.

5. **True or False (1 point):** The Peterson's solution to the critical section problem does not ensure *bounded waiting*.

6. **Select the correct answer (1 point):** The goal of using *thread pools* is to:
(a) make is easier for the programmer to write correct multi-threaded programs
(b) reduce the expense of thread creation and removal

7. **Answer the following (2 points):** Given the definition of the `TestandSet` instruction, use the instruction in the lock() function to achieve mutual exclusion.

```
boolean TestandSet (boolean *target)
{
   boolean rv = *target;
   *target = TRUE;
   return rv:
}
```

```
void lock(int *mutex)
{
   while (TestandSet(mutex))
      ;
}
```

8. **Answer the following (3 points):** In the below code snippets with one *producer* and one *consumer*, the variables, `counter` and `buffer` are shared. Circle the **critical regions** of code in the producer.

```
        Producer                          Consumer
        --------                          --------
/* wait if buffer full */          /* wait if buffer empty */
while (counter == 10);             while (counter == 0);

/* produce data */                 /* consume data */
buffer[in] = idata;                odata = buffer[out];
in = (in+1) % 10;                  out = (out+1) % 10;

/* update items in buffer*/        /* update items in buffer */
counter++;                         counter--;
```

-1

-1

+3

total; no one gets < 0