1. [12 marks total; 1 mark for each]
    a. TRUE
    b. FALSE
    c. TRUE
    d. TRUE
    e. TRUE
    f. FALSE
    g. TRUE
    h. FALSE
    i. TRUE
    j. TRUE
    k. FALSE
    l. TRUE

2. [14 marks total]
   a. [3 marks total] one mark for each correct point

      1. *Multi-threaded code lacks mutual exlusion/resource protection.*
      2. *Compiler optimizations (register alloc., re-ordering)*
      3. *CPU LW/SW reordering*

   b. [2 marks total] one mark for indicating NO, one mark for explanation

      No.
      Two threads executing on different CPUs can still access the same data at the same time.

   c. [2 marks total] one mark for indicating NO, one mark for explanation

      No.
      Each thread accesses a unique element of the array, so no two threads will be attempting access to the same entry.

d. [5 marks total] one mark for each point; students may use code here

1. *assert lock & cv are not NULL*
2. *Acquire wchan lock (wchan_lock(cv->wchan))*
3. *Release lock (lock_release(lock))*
4. *Sleep on wchan (wchan_sleep(cv->wchan))*
5. *Acquire lock (lock_acquire(lock))*

e. [2 marks total] one mark for each point

Binary semaphores can be released by any thread.

Binary semaphores must be used correctly, else multiple threads may enter the critical section if V is called more than P.

Also accept: inability to solve priority inversion problem

3. [4 total marks] -0.5 marks for each mistake

Full Marks Solution 1

USER STACK	KERNEL STACK
Application code	trapframe
waitpid	mips_trap
	syscall
	sys_waitpid
	trapframe
	mips_trap
	mainbus_interrupt
	timer_interrupt_handler

Full Marks Solution 2

USER STACK	KERNEL STACK
Application code	trapframe
waitpid	Interrupt handler
	syscall
	sys_waitpid
	trapframe
	Interrupt handler
	timer_interrupt_handler

4. [5 total marks]
   a. [1 mark total] one mark for correct answer

      Yes.

   b. [4 marks total] marks as indicated below

      [1 mark] add a lock or binary semaphore (with initial value 1)
      [2 marks] acquire lock/binary semaphore after P(cars) and P(parkingSpots)
      [1 mark] release lock/binary semaphore before V(cars) and V(parkingSpots)

5. [9 marks total]
   a. [6 marks total] one mark for each point

      1. *Check if the new size is less than the max seg size, or return error.*
      2. *Check if there is enough physical memory for the new, larger segment, or return error.*
      3. *Allocate the new segment in physical memory.*
      4. *Copy the old segment contents to new segment.*
      5. *Update relocation/limit values for the process*
      6. *Update MMU relocation/limit registers*

   b. [3 marks total] one mark for each point

      1. *External fragmentation*
      2. *Complexity of segment growth/shrinkage*
      3. *No support for paging*

6. [12 total marks]
   a. [1 mark total] one mark for correct answer

   12

   b. [1 mark total] one mark for correct answer

   2^64/2^12 = 2^52

   c. [1 mark total] one mark for correct answer

   2^16/2^12 = 2^4

   d. [1 mark total] one mark for correct answer

   52

   e. [1 mark total] one mark for correct answer

   4

   f. [1 mark total] one mark for correct answer

   2^64/2^16 = 2^48

   g. [1 mark total] one mark for correct answer

   3KB

h. [4 marks total] one mark for each correct answer

1. *YES*
2. *NO*
3. *NO*
4. *YES*

i. [1 mark total] one mark for correct answer

2^4 pages * 2^4 PTE size = 2^8 bytes

7. [6 total marks]
   a. [1 mark total] one mark for correct answer

      N/2

   b. [1 mark total] one mark for correct answer

      Yes.

   c. [4 marks total] marks as indicated

      NM locks -> one for each element of each queue. ***[1 mark]***

      The idea is, you acquire the lock of queueAelementM and queueBelementN ***[2 marks]*** *protecting that exact spot in memory, but permitting actions on the rest of the queues by other threads.*

      NM/2 threads can execute in parallel this way. ***[1 mark]***

      ALTERNATE SOLUTIONS MAY EXIST, if they work, give FULL marks.

8. [6 total marks] marks as indicated

```
try_acquire( lock lk )
   spinlock_acquire( lk->spin ) [1 mark]
       If ( lk->held )  [1 mark]
           spinlock_release( lk->spin) [1 mark]
           Return false; [1 mark]
      Else
           lk->held = true  [1 mark for acquiring the lock]
          lk->owner = curthread
          spinlock_release( lk->spin ) [1 mark]
```

ACCEPT ANY PSEUDOCODE OR LIST THAT WORKS.

9. [8 total marks] one mark for each correct segment number, one mark for each correct physical address

ADDR	SEGMENT	PHYS ADDR
0x0EA5 EE00	0	exception
0x0000 0ACE	0	0x1000 0ACE
0x3000 00C5	0	exception
0x2000 AFAF	0	exception