Quiz 2
Question 1

- Suppose that Nachos had Monitors instead of Semaphores. Show how you could implement Locks and Condition Variables using Monitors. Show the actual monitor definitions and the code for the Locks (but not the code for the condition variables). Use C++ class syntax for your monitor definition.
Lock Monitor Definition

monitor Lock {
    public:
        void Acquire();
        void Release();

    private:
        bool busy;  // Initially false
        condition waiting;
};
Lock Code

Lock::Acquire()
{
    if(busy)
        waiting.wait();

    busy = TRUE;
}

Lock::Release()
{
    busy = FALSE;
    waiting.signal();
}
Condition Variable Definition

monitor Condition {
    public:
        void Wait();
        void Signal();
        void Broadcast();
    
    private:
        int numwaiting;
        Condition waiting;
};
Condition Variable Code

Condition::Wait()
{
    num_waiting++;  
    waiting.wait();  
}

Condition::Signal()
{
    if(num_waiting) {
        num_waiting--;  
        waiting.signal();  
    }
}
More Condition Variable Code

Condition::Broadcast()
{
    while(num_waiting) {
        num_waiting--;
        waiting.signal();
    }
}
Question 2

• Explain the difference between synchronous and asynchronous IPC mechanisms. Explain what effect switching from synchronous to asynchronous operations could have on the turnaround time of a process using them.
Synchronous IPC

• Synchronous IPC mechanisms cause the calling process to block while waiting for the operation to complete.
• The IPC function call doesn't return until it has completed the operation.
Asynchronous IPC

- Asynchronous IPC mechanisms allow the calling process to return immediately and may complete the operation later.
Turnaround time

• The turnaround time with asynchronous IPC could get smaller, since the process would not have to wait for IPC calls to finish but could instead keep running.
Question 3

• List and very briefly describe the four conditions that must hold for deadlock to occur.
Conditions

• Mutual Exclusion - The resources must be accessed by only one process at a time
• Hold and Wait - a process must hold resources while waiting for other resources
• Circular Wait - a set of processes must each hold something that the others need
• No Preemption - processes must not be able to steal resources from each other
Question 4

- Consider a hardware design with 2 reference bits instead of 1. Explain how we can implement a better approximation to LRU.
Solution

• Use one of the bits as a reference bit just like the single bit case.

• Use the other bit as a "referenced last time" bit. In other words, every time we go to reset the reference bit to zero, copy its value to the "referenced last time" bit.

• Now, if both bits are zero then the page hasn't been used for a long time. If the reference bit is zero and the "referenced last time" bit is 1, then the page was used a while ago, and if the reference bit is zero then the page was used quite recently.