Answer *any* five problems.

**Problem 1**: (20 pts)
A joint bank account is being accessed by 3 of its co-owners at 3 different ATMs. The account has $10,000 in savings, and $500 in checking. One user transfers $1000 from savings to checking; another deposits $500 in checking; the third withdraws $500 from checking. The code for the three operations in the ATM controllers is:

```java
withdraw(source, amount) {
    money[source] = money[source] - amount;
    dispense(amount); //produce bills to customer
}

deposit(destination, amount) {
    receive(amount); //mechanical receive of the envelope
    money[destination] = money[destination] + amount;
}

transfer(source, destination, amount) {
    money[destination] = money[destination] + amount;
    money[source] = money[source] - amount;
}
```

- (a) (10) What are the possible final values for the checking and savings accounts?
- (b) (10) Can you protect against this inconsistency? Suggest an implementation (complete serialization is unacceptable since there are so many customers using these functions); is deadlock a problem?

**Problem 2**: (20 pts)
You’ve just been hired by Mother Nature to help her out with the chemical reaction to form water, which she doesn’t seem to be able to get right due to synchronization problems. The trick is to get two H atoms and one O atom all together at the same time. The atoms are threads. Each H atom invokes a procedure `hReady` when it’s ready to react, and each O atom invokes a procedure `oReady` when it’s ready. For this problem, you are to write the code for `hReady` and `oReady`. The procedures must delay until there are at least two H atoms and one O atom present, and then one of the procedures must call the procedure `makeWater` (which just prints out a debug message that water was made). After the `makeWater` call, two instances of `hReady` and one instance of `oReady` should return. Write the code for `hReady` and `oReady` using either semaphores or locks and condition variables for synchronization. Your solution must avoid starvation and busy-waiting.
Problem 3: (20 pts)

- (a) (4) A computer system has $d$ tape drives for which 10 processes compete. Each process may request up to 2 drives. For what values of $d$ is the system deadlock free?

- (b) (8) What value of $d$ would guarantee the same for a system of $n$ processes each requesting up to $r$ drives ($r < d$)?

- (c) (8) A potential deadlock management technique is to choose $n$ such that the formula you developed in (b) holds for every resource. Discuss two more liberal schemes for managing deadlock situations than enforcing the limit above. Demonstrate (using examples if necessary) that they are more liberal.

Problem 4: (20 pts)

Which of the following events can lead from a safe to unsafe state; which can lead from a deadlock-free to a deadlocked state: (a) A process finishes (releasing its resources); (b) a process increases its claim; (c) A new process comes in (with a non-zero claim vector); (d) A process makes a request within its claim. Explain (be specific, use examples if necessary).

Problem 5: (20 pts - Token open-ended question)

Because of your recent successes in the scheduling group in Chimera, you have been promoted to the head of the memory management group. As a first step, you are entertaining proposals on how to structure the memory from you team-members. One proposal suggests having fixed partitions with each process being allocated exactly one partition. Each process is free to manage its partition any way it feels is best for it (simple or virtual segments or pages, buddy system, fixed/dynamic partitions). Comment on the advantages/disadvantages of this solution. It might help to consider some specific organizations and what support would be necessary for them. Be concise and precise.

Problem 6: (20 pts)

- (a) (8) Consider an inverted page table scheme with the hash table pinned (always in memory). The details of the hashing function are unimportant; chaining is used if two values hash into the same location. Explain how a page number is translated using this scheme; what is the maximum number of memory references necessary?

- (b) (12) Explain in detail what happens when a memory reference occurs on a system with virtual memory, a TLB and a cache using an inverted page table. If access to the page table is necessary, reference your answer in part a (do not explain it again). What is the maximum number of page faults possible? What is the maximum number of memory references? (Note: you can answer this part even if you couldn't get part (a))