Problem 1
Consider the following program:

```c
int buffer[M];
int nextIn = 0, nextOut = 0;
Semaphore: freeSlot = M, notEmpty = 0, enter = 1;
Producer: do {
    ... Produce inItem ...
    Wait(freeSlot); Wait(enter); // Interchange 1
    buffer[nextIn] = inItem; // Put new Item
    nextIn = (nextIn + 1) mod M;
    Signal(enter); Signal(notEmpty); // Interchange 2
} until (DONE);
Consumer:  do {
    Wait(notEmpty); Wait(enter); // Interchange 3
    outItem = buffer[nextOut]; // Get Item
    nextOut = (nextOut + 1) mod M;
    Signal(enter); Signal(freeSlot); // Interchange 4
    ... Consume outItem ...
} until (DONE);
```

a) Describe the purpose of each semaphore.

b) How would the meaning of the original program change if the pair of statements labeled Interchange 1 were executed in reverse order (interchanged)? That is, Wait(enter); Wait(freeSlot); instead of Wait(freeSlot); Wait(enter).

c) How would the meaning of the original program change if the pair of statements labeled Interchange 2 were executed in reverse order (interchanged)?

d) How would the meaning of the original program change if the pair of statements labeled Interchange 3 were executed in reverse order (interchanged)?

Problem 2
Look at Stallings, Problem 5.11 and answer the following questions.

a) Briefly describe the main idea(s) that you will use in the algorithm that will provide the access guarantee described in this problem.

b) Give the algorithm based on testset.

Problem 3
Stallings, Problem 5.13.