TUTORIAL FIVE

Process Synchronization- Part II

1. Consider a semaphore S that uses blocking implementation. Suppose at the current time instant, a) 2 processes are holding S having successfully executed Wait(S) previously, and b) the blocked process list for S contains 4 Process Control Blocks (PCBs).

   a) What is the value of S at the current time instant? S = -4

   b) Suppose the value of S changes to -1 after executing **Signal(S) five times and Wait(S) two times** in some order. What is the minimum intermediate value that S can have during these operations? Justify your answer. S = -6

   c) What is the largest value that S can ever have? Justify your answer. S = 2

2. Alice is playing a game that involves pairing apples with oranges. The game produces apples and oranges in separate baskets at random instants. For Alice to get points, she must **pair exactly 1 apple with 2 oranges** by picking them from the respective baskets. Each basket is protected by a semaphore; A for the apple basket and O for the orange basket. To access a basket, Alice must first acquire the corresponding semaphore.

   Complete the code in the below figure to help Alice to play this game. You should only use the operations Wait(A), Wait(O), Signal(A) and Signal(O) in the boxes provided. You may use any number of these operations (including zero) in each of the boxes. **Your solution must be deadlock-free even in the presence of other players who may be executing a different code**.

```
[ 1 ]                                   Wait(A);
// Pick 1 apple
[ 2 ]                                   Signal(A); Wait(O);
If (atleast 2 oranges in basket) {
** Pick 2 oranges and pair with apple. **
[ 3 ]                                   Signal(O);
} else {
[ 4 ]                                   Signal(O); Wait(A);
** Drop 1 apple back in basket **
[ 5 ]                                   signal(A);
}
```

3. Describe how *Wait*(*S*) and *Signal*(*S*) of a semaphore can be implemented using a *TestAndSet* instruction, given the below semaphore structure definition.
   Hint: *The semaphore value and the process queue L are shared variables among different processes when those processes access the semaphore.*

```
typedef struct {
    int value;
    struct process *L;
} semaphore;
```

```
Wait(S) {
while (TestAndSet(lock));
S.value--;
if (S->value < 0) {
    add this process to S->list;
    lock = false;
    block();
}
lock = false;
}
```

same for signal.

1b) The minimum will be -6.
If none of the processes signal(S) before 2 wait(S) have been called.

1c) S=2 is the largest S. As two processes cussently acquire S and others are blocked.