CS 2006 Operating Systems

Assignment 3 Q2 and Q3

Roll No : 22L-6552

Section : 4H

Name : Faizan Shabir

22L-6552 Faizan Shabir 4H

Q2) This question maps to the bounded buffer problem.

```
class Stack
{
  private:
          int * a; // array of stack
          int max; // max size of array
          int top; // stack top

          Semaphore full; //initialized in constructor
          Semaphore empty; //initialized in constructor

          Semaphore mutex; //initialized in constructor

  public:
          Stack (int m)
          {
               a = new int [m];
               max = m;
               top = 0;

               full = 0; //Semaphore initialized

               empty = max; // Semaphore initialized

               mutex = 1; // Semaphore initialized
          }
void push (int x)
{
        wait (empty); //proceed only if stack
                        has empty slots
        wait (mutex); // acquire lock

        a[top] = x; // execute critical
        ++top;               section

        signal (mutex); // release lock

        signal (full); // one ~~[illegible]~~ slot of the
                        stack is now full.
}

int pop ()
{
        wait (full); // proceed only if stack
                       has atleast 1 full slot
        wait (mutex); // acquire lock

        int temp = top; // execute critical
        --top;                 section

        signal (mutex); // release lock

        signal (empty); // one slot of the
                          stack is now empty
        return a[temp];
}
};
```

22L-6552 Faizan Shabir 4H

Q3) This question maps to the readers writers problem.

```
// Semaphore initializations
Semaphore enterMutex = 1;
Semaphore comedyMutex = 1;
Semaphore drama_Mutex = 1;
// Counter initializations
int comedyCount = 0;
int dramaCount = 0;

while (1) // Comedy process
{
      wait (comedyMutex); // comedy-fan-wants-to-enter
      comedyCount++;
      if (comedyCount == 1)
      {
            wait (enterMutex);
      }
      signal (comedyMutex);
      // Enter the section
      wait (comedyMutex); // comedy-fan-leaves
      comedyCount--;
      if (comedyCount == 0)
      {
            signal (enterMutex);
      }
      signal (comedyMutex);
}
```

```
while(1) // drama process
{
    wait(dramaMutex); //drama-enthusiast-wants-to
    dramaCount++;                          enter
    if (dramaCount == 1)
    {
        wait(enterMutex);
    }
    Signal(dramaMutex);

    // Enter the section

    wait(dramaMutex); // drama-enthusiast-leaves
    dramaCount--;
    if (dramaCount == 0)
    {
        Signal(enterMutex);
    }
    signal(dramaMutex);
}
```