Using an analogy of cakes being aded to a window display:

Item buffer [n]; - This is a pool of n buffers - each can hold 1 item. This represents the window display capacity - there are n stands which can hold 1 cake each
Semphore mutex = 1; - This provides mutually exclusive access to the buffer pool. Only 1 person has access to the cake display at a time
Sempahore full = 0; - This semaphore counts the number of full buffers (stands with cakes on) in the window
Semaphore empty = n; - Similarly this semaphore keeps track of the number of empty cake stands
— — —

Semaphores are variables that control access to common resources. The basic code works like:

```
wait(Semaphore) {.     // 'Wait' implemented when something wants to use the resource
      while( S < = 0 ) ;  // if the semaphore (resource) is empty (or can be a boolean false - not available)  - get stuck waiting in loop
      S - - ;              // If semaphore isn't empty (eg at the start or after someone has incremented it (replenished it)
}                            then you now have access to the resource and you have to decrement the semaphore
                             to indicate you are using it now.

signal(semaphore) {  // 'Signal' is implemented when you are finished using a resource - you increment the semaphore
                        so someone else can use the resource
      S++;
}
```

Producer: A person who makes the cakes and adds them to the stands

```
 do {
        //  make a cake
      wait(empty);       // ask for an empty cake stand ( and then decrement the number of empty ones when one is free)
      wait(mutex);       // ask for access to the cake display ( and decrement semaphore when access is available - so now you have access)

        //  add cake to a stand
     signal(mutex);    // signal that you're done with the display an increment semaphore so someone else can access it
     signal(full);       // signal that you've added 1 cake to one of the display stands so increment the number of full stands

} while (TRUE);
```

Consumer: Takes cakes of the stands and eats them:

```
 do {
      wait (full);       // ask for a cake ( wait for a 'full' cake stand to be available in the display)
      wait (mutex) ;  //ask for access to the window display

        //  remove cake from a stand
      signal(mutex);     // Signal that you're done with the display
      signal(empty);     // Signal that you've emptied one cake stand now - so increment the number of empty ones

       //  eat the cake 😃

} while (TRUE);
```