**Design: Hotel Simulation**

Here is an example design for the above project:

**(1) List of Semaphores and their Purpose:**

mutex: This semaphore is used to ensure that only one thread at a time can access the shared resource, which is the printer. Its initial value is 1.

printQueue: This semaphore is used to represent the number of available slots in the print queue. Its initial value is 10, which is the maximum number of documents that can be queued for printing at any given time.

docQueue: This semaphore is used to represent the number of available slots in the document queue. Its initial value is 10, which is the maximum number of documents that can be queued for processing by the printer at any given time.

**(2) Pseudocode for each function:**

a. DocumentGenerator():

while true:

create a new document

wait(docQueue)

wait(mutex)

put the new document in the document queue

signal(mutex)

signal(printQueue)

b. Printer()

while true:

wait(printQueue)

wait(mutex)

remove a document from the document queue

print the document

signal(mutex)

signal(docQueue)

The preceding pseudocode is comparable to the one used for the barbershop problem in the textbook. In order to access the shared resource—the queues—in both functions, we must first obtain the required semaphores. The necessary action is then carried out, either by adding a new document to the document queue in DocumentGenerator() or by removing an existing document from the document queue and printing it in Printer (). In order for additional threads to use the shared resource, we finally release the acquired semaphores.

In DocumentGenerator(), we first call wait to watch for a spot to open up in the document queue (docQueue). We obtain the mutex semaphore as soon as a slot opens up to guarantee exclusive access to the document queue. After that, we release the mutex semaphore and add the new document to the document queue. Lastly, we notify the Printer() thread that a new document is ready for processing by signaling the printQueue semaphore.

We first call wait in Printer() to wait for a document to become available in the document queue (printQueue). We obtain the mutex semaphore as soon as a document becomes available to guarantee exclusive access to the document queue. The document is then deleted from the document queue, printed, and the mutex semaphore is released. In order to inform the DocumentGenerator() thread that there is a space available for a new document to be added to the document queue, we finally signal the docQueue semaphore.

Overall, the design described above makes sure that multiple threads can access the queues, a shared resource, in a synchronized way, preventing race conditions and deadlocks. In order to avoid any data inconsistencies, semaphores are used to make sure that the threads acquire and release the required resources in the proper sequence and at the appropriate times.