1. Servers can be designed to limit the number of open connections. For example, a server may wish to have only *N* socket connections at any point in time. As soon as *N* connections are made, the server will not accept another incoming connection until an existing connection is released. Explain how semaphores can be used by a server to limit the number of concurrent connections.

Answer: A semaphore is initialized to the number of allowable open socket connections. When a connection is accepted, the acquire()method is called, when a connection is released, the release() method is called. If the system reaches the number of allowable socket connections, subsequent calls to acquire() will block until an existing connection is terminated and the release method is invoked.

Write a bounded-buffer monitor in which the buffers (portions) are embedded within the monitor itself.

Answer:

```
monitor bounded_buffer {
      int items[MAX_ITEMS];
      int numItems = 0;
      condition full, empty;
      void produce(int v) {
            while (numItems == MAX_ITEMS)
            full.wait();
            items[numItems++] = v;
            empty.signal();
      }
      int consume() {
            int retVal;
            while (numItems == 0)
                  empty.wait();
            retVal = items[--numItems];
            full.signal();
            return retVal;
      }
}
```

2. A file is to be shared among different processes, each of which has a unique number. The file can be accessed simultaneously by several processes, subject to the following constraint: The sum of all unique numbers associated with all the processes currently accessing the file must be less than *n*. Write a monitor to coordinate access to the file.

Answer: The pseudocode is as follows:

```
monitor file access {
int curr sum = 0;
int n;
condition c;

void access file(int my num)
{
      while (curr sum + my num >= n)
      c.wait();
      curr sum += my num;
}
void finish access(int my num) {
      curr sum -= my num;
      c.broadcast();
}
```