Interpretation #1 - All semaphore key requests pass through a FIFO.

Clarification paragraph to 14.2.3 get():
The semaphore waiting queue is First-In First-Out (FIFO). This does not guarantee the order in which processes arrive at the queue, only that their arrival order shall be preserved by the semaphore and a blocked semaphore will block subsequent semaphore requests until the first semaphore's request is satisfied, even if there are enough keys to immediately satisfy the request from a later semaphore request.

Use model - user assumes that proc_a, proc_b and proc_d can execute in parallel but proc_c must execute alone.

```
module sematest1;
  semaphore bucket=new(3);

  initial begin : proc_a
       bucket.get;      // t0 - 2 keys available
    #4 bucket.put;      // t4 - 2 keys available
                        // proc_c & proc_d still blocked
    #2 bucket.get;      // t6 - proc_a blocked behind proc_d
                        // t7 - proc_a unblocks - continue
    #2 bucket.put;      // t9 - 3 keys available
    #1 $finish;         // t10 - finish simulation
  end

  initial begin : proc_b
    #1 bucket.get;      // t1 - 1 key available
    #4 bucket.put;      // t5 - 3 keys available
                        // proc_c unblocks, proc_d blocked
  end

  initial begin : proc_c
    #2 bucket.get(3);   // t2 - blocked - only 1 key
                        // t5 - proc_c unblocks - continue
    #2 bucket.put(3);   // t7 - 3 keys in bucket
                        // proc_d & proc_a unblock
                        // 1 key available
  end

  initial begin : proc_d
    #3 bucket.get;      // t3 - blocked behind proc_c
                        // t7 - proc_d unblocks - continue
    #1 bucket.put;      // t8 - 2 keys available
  end
endmodule
```

Interpretation #2 - A semaphore can get its required keys even if another semaphore is blocked waiting for additional unavailable keys. LRM Clarification definitely needed!!

Use model - processes that request fewer keys will have priority over processes that request more keys.

Question: What will happen if there are no keys, process-A requests two keys (is put on the requesting FIFO) and later process-B requests one key?

Is process-B put on the requesting FIFO behind process-A??

If one key is returned, does process-B come off the not-so-FIFO and become active??

I can live with Interpretation #2, but the wording is clearly inadequate to describe this behavior.

```
module sematest2;
  semaphore bucket=new(3);

  initial begin : proc_a
       bucket.get;      // t0 - 2 keys available
    #4 bucket.put;      // t4 - 1 key available
                        // proc_c still blocked
    #2 bucket.get;      // t6 - 1 key available
                        // proc_c still blocked
    #2 bucket.put;      // t8 - 3 keys available
                        // proc_c unblocks
    #3 $finish;         // t11 - finish simulation
  end

  initial begin : proc_b
    #1 bucket.get;      // t1 - 1 key available
    #4 bucket.put;      // t5 - 2 keys available
                        // proc_c still blocked
  end

  initial begin : proc_c
    #2 bucket.get(3);   // t2 - blocked - only 1 key
                        // t8 - proc_c unblocks - continue
    #2 bucket.put(3);   // t10 - this statement does not
                        // execute until #2 after get(3)
                        // unblocks (at time t8)
                        // 3 keys available
  end

  initial begin : proc_d
    #3 bucket.get;      // t3 - 0 keys available
                        // proc_c still blocked
```

```
    #4 bucket.put;      // t7 - 2 keys available
                        // proc_c still blocked
  end
endmodule

------------------------------------------------------------
WORKAROUND FOR Interpretation #1 - To mimic the behavior that all
semaphore key requests pass through a FIFO.

Make requests using the repeat command

Use model - user assumes that proc_a, proc_b and proc_d can
execute in parallel but proc_c must execute alone.
------------------------------------------------------------

module sematest3;
  semaphore bucket=new(3);

  initial begin : proc_a
       bucket.get;      // t0 - 2 keys available
    #4 bucket.put;      // t4 - proc_c takes another key
                        // proc_c & proc_d still blocked
    #2 bucket.get;      // t6 - proc_a blocked behind proc_d
                        // t7 - proc_a unblocks - continue
    #2 bucket.put;      // t9 - 3 keys available
    #1 $finish;         // t10 - finish simulation
  end

  initial begin : proc_b
    #1 bucket.get;      // t1 - 1 key available
    #4 bucket.put;      // t5 - proc_c takes a 3rd key
                        // proc_c unblocks, proc_d blocked
  end

  initial begin : proc_c
    #2 repeat(3) bucket.get;  // t2 - only 1 key acquired
              // proc_c blocked waiting for two more keys
                        // t5 - proc_c unblocks - continue
    #2 bucket.put(3);   // t7 - 3 keys in bucket
                        // proc_d & proc_a unblock
                        // 1 key available
  end

  initial begin : proc_d
    #3 bucket.get;      // t3 - blocked behind proc_c
                        // t7 - proc_d unblocks - continue
    #1 bucket.put;      // t8 - 2 keys available
  end
endmodule

------------------------------------------------------------
```