Name: Varun Lingabathini ID:026626468

CECS 526 Assignment 3 (2 point)
Due : February 18, 2020, by class time on BeachBoard

There are N blocks of storage. Initially, these blocks are empty and linked together on "freelist" (shown in the table below as •-•-•-• for N=4). Three processes (Process 1, Process 2, and Process 3) communicate through the use of these blocks as depicted in the code on next page. Note that all three processes execute infinite loops. Denoting the execution of an iteration of the loop in Process 1, Process 2, and Process 3 as P1, P2, and P3, respectively, and supposing we have observed the following order of arrivals: P3, P1, P2, P1, P1, P1, and letting N=4, show in the table below all changes to the variables listed as the iterations execute, and the final values in those variables when all processing due to the above iterations' execution completes.

Process being executed	freelist	nFL		list1	nL1		list2	nL2	
		value	queue		value	queue		value	queue
Initially	•-•-•-•	4	empty	empty	0	Empty	empty	0	empty
P3	•-•-•-•	4	Empty	Empty	0	Empty	empty	0	P3
P1	•-•-•	3	Empty	•	1	Empty	empty	0	P3
P2	•-•-•	3	Empty	Empty	0	Empty	•	1	empty
P3	•-•-•-•	4	Empty	Empty	0	Empty	empty	0	empty
P1	•-•-•	3	Empty	•	1	Empty	empty	0	empty
P1	•-•	2	Empty	•-•	2	Empty	empty	0	empty
P1	•	1	Empty	•-•-•	3	Empty	empty	0	empty
None	•	1	Empty	•-•-•	3	Empty	empty	0	empty

/* Three processes communicate through blocks of data. Initially “freelist” contains N free blocks, while “list1” and “list2” are both empty. */

```
/* Shared Variables */
   var mutexFL, mutexL1, mutexL2 : semaphore := 1, 1, 1;
       nFL, nL1, nL2 : semaphore := N, 0, 0;
       freelist, list1, list2 : pointer to block : head_of_list_of_N_blocks,empty,empty;

/* Process 1 */
   var b : block;
   while true do
       begin
          SP (mutexFL,1,1; nFL,2,1);      /* make sure at least 2 blocks in freelist */
             b := unlink (freelist);      /* remove a block from freelist */
          SV (mutexFl,1);
          produce info in block b;
          SP (mutexL1,1,1);
             link (b, list1);             /* link block b to list1 */
          SV (mutexL1,1; nL1,1);
       end

/* Process 2 */
   var x, y : block;
   while true do
       begin
          SP (mutexL1,1,1; mutexFL,1,1; nL1,1,1; nFL,1,1);  /* make sure blocks exist in
                                                               both list1 & freelist */
             x := unlink (list1);  y := unlink (freelist);  /* remove a block from each
                                                               of list1 & freelist   */
          SV (mutexL1,1; mutexFl,1);
          use block x & produce info in block y;
          SP (mutexFL,1,1);
             link (x, freelist);          /* link block x to freelist */
          SV (mutexFL,1; nFL,1);
          SP (mutexL2,1,1);
             link (y, list2);             /* link block y to list2 */
          SV (mutexL2,1; nL2,1);
      end

/* Process 3 */
   var c : block;
   while true do
       begin
          SP (mutexL2,1,1; nL2,1,1);      /* make sure blocks exist in list2 */
             c := unlink (list2);         /* remove a block from list2 */
          SV (mutexL2,1);
          consume info in block c;
          SP (mutexFL,1,1);
             link (c, freelist);          /* link block c to freelist */
          SV (mutexFL,1; nFL,1);
      end
```