PONTIFICIA UNIVESIDAD CATÓLICA DEL PERÚ
ESCUELA DE POSGRADO
MAESTRÍA EN INFORMÁTICA

INF646 MÉTODOS FORMALES
Examen 1
2016 – 2

Con respuestas

Prepare un directorio de trabajo con el nombre <*su-código-de-8-dígitos*>.
Este directorio es para desarrollar los programas de las preguntas del examen. Los nombres de los programas se indican en las preguntas.
Las respuestas a las preguntas y su comentarios usted puede preparar en el archivo <*su-código-de-8-dígitos*>**.txt**.
Al final del examen, comprime todo el directorio de trabajo al archivo <*su-código-de-8-dígitos*>**.zip** y colóquelo en la carpeta **Documentos del curso/Examen 1/Buzón/** en el Campus Virtual.

A esta hoja están acompañando los 4 archivos: **lbs_3.6.4a.pml**, **lbs_3.7.1a.pml**, **lbs_3.7.3a.pml**, y **zeroA.pml**. Cópialos a su directorio de trabajo.

Pregunta 1. (10 puntos – 90 min.) (*The Little Book of Semaphore (2.2.1) by A. Downey, pp.29-37*)
La solución de una barrera según el código de la sección 3.6.4:

```
$ cat -n lbs_3.6.4a.pml | expand
     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.6 Barrier
     7      3.6.4 Barrier solution
     8  */
     9
    10  #define N 3
    11
    12  #define wait(sem)   atomic { sem > 0; sem-- }
    13  #define signal(sem) sem++
    14
    15  byte count=0, mutex=1, barrier=0 /* barrier is locked */
    16
    17  proctype P(byte i) {
    18      do
    19      ::  wait(mutex)
    20              count++
    21          signal(mutex)
    22
    23          if
    24          :: count == N ->
    25                          signal(barrier)
    26          :: else
    27          fi
    28
    29          wait(barrier)    /* turnstile pattern */
    30          signal(barrier)
    31
    32          break    /* one only iteration */
```

```
    33      od
    34  }
    35
    36  init {
    37      byte i
    38
    39      atomic {
    40          for (i: 1 .. N) {
    41              run P(i)
    42          }
    43      }
    44      _nr_pr == 1 ->
    45                      assert(barrier != 0) /* barrier (turnstile) is open! */
    46  }
```

```
$ spin -run lbs_3.6.4a.pml | expand > lbs_3.6.4a.pan_out
$ cat lbs_3.6.4a.pan_out

(Spin Version 6.4.5 -- 1 January 2016)
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +

State-vector 48 byte, depth reached 40, errors: 0
      498 states, stored
      465 states, matched
      963 transitions (= stored+matched)
       12 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
      0.036          equivalent memory usage for states (stored*(State-vector +
overhead))
    0.289       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  128.730       total actual memory usage


unreached in proctype P
        (0 of 19 states)
unreached in init
        (0 of 13 states)

pan: elapsed time 0 seconds
```

La verificación con la búsqueda de violaciones de asertos y estados finales inválidos no encuentra errores.

a) (`lbs_3.6.4b.pml`) (3 puntos – 27 min.) ¿Cuál es el valor final de la variable **`barrier`** para *N* procesos? Prepare el modelo correspondiente en el archivo **`lbs_3.6.4b.pml`**, presente los resultados del analizador en el archivo **`lbs_3.6.4b.pan.out`**, y, para cada caso encontrado, los resultados de simulación en el archivo **`lbs_3.6.4b.pml`*i*`.trail.out`** donde *i* corresponde al número del caso.
Respuesta: el valor final de la variable `barrier` es 1..*N*.

```
$ cat -n lbs3.6.4b.pml | expand
...
      45                    assert(barrier == 0) /* find all scenarios */
```

```
...

$ spin -run -e lbs_3.6.4b.pml | expand > lbs_3.6.4b.pan.out

$ cat lbs_3.6.4b.pan.out
pan:1: assertion violated (barrier==0) (at depth 36)
pan: wrote lbs_3.6.4b.pml1.trail
pan: wrote lbs_3.6.4b.pml2.trail
pan: wrote lbs_3.6.4b.pml3.trail

(Spin Version 6.4.5 -- 1 January 2016)
       + Partial Order Reduction

Full statespace search for:
       never claim               - (none specified)
       assertion violations      +
       cycle checks        - (disabled by -DSAFETY)
       invalid end states  +

State-vector 48 byte, depth reached 40, errors: 3
      498 states, stored
      465 states, matched
      963 transitions (= stored+matched)
       12 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.036    equivalent memory usage for states (stored*(State-vector + overhead))
    0.289    actual memory usage for states
  128.000    memory used for hash table (-w24)
    0.534    memory used for DFS stack (-m10000)
  128.730    total actual memory usage


unreached in proctype P
       (0 of 19 states)
unreached in init
       (0 of 13 states)

pan: elapsed time 0 seconds

$ spin -t1 lbs_3.6.4b.pml | expand > lbs_3.6.4b.pml1.trail.out

$ cat lbs_3.6.4b.pml1.trail.out
spin: lbs_3.6.4b.pml:45, Error: assertion violated
spin: text of failed assertion: assert((barrier==0))
spin: trail ends after 37 steps
#processes: 1
                count = 3
                mutex = 1
                barrier = 1
 37:    proc  0 (:init::1) lbs_3.6.4b.pml:46 (state 13) <valid end state>
4 processes created

$ spin -t2 lbs_3.6.4b.pml | expand > lbs_3.6.4b.pml2.trail.out

$ cat lbs_3.6.4b.pml2.trail.out
spin: lbs_3.6.4b.pml:45, Error: assertion violated
spin: text of failed assertion: assert((barrier==0))
spin: trail ends after 38 steps
#processes: 1
                count = 3
                mutex = 1
```

```
                barrier = 2
 38:    proc  0 (:init::1) lbs_3.6.4b.pml:46 (state 13) <valid end state>
4 processes created

$ spin -t3 lbs_3.6.4b.pml | expand > lbs_3.6.4b.pml3.trail.out

$ cat lbs_3.6.4b.pml3.trail.out
spin: lbs_3.6.4b.pml:45, Error: assertion violated
spin: text of failed assertion: assert((barrier==0))
spin: trail ends after 39 steps
#processes: 1
                count = 3
                mutex = 1
                barrier = 3
 39:    proc  0 (:init::1) lbs_3.6.4b.pml:46 (state 13) <valid end state>
4 processes created
```

b) (`lbs_3.7.1a.pml`) (**3 puntos – 27 min.**) En el archivo **`lbs_3.7.1a.pml`**, está el modelo correspondiente al código de la sección 3.7.1:

```
$ cat -n lbs_3.7.1a.pml | expand
     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.7 Reusable barrier
     7      3.7.1 Reusable barrier non-solution #1
     8  */
     9
    10  #define N 3
    11
    12  #define wait(sem)   atomic { sem > 0; sem-- }
    13  #define signal(sem) sem++
    14
    15  byte count=0, mutex=1, turnstile=0 /* turnstile is locked */
    16
    17  proctype P(byte i) {
    18      do
    19      ::  wait(mutex)
    20              count++
    21          signal(mutex)
    22
    23          if
    24          :: count == N ->
    25                          signal(turnstile)
    26          :: else
    27          fi
    28
    29          wait(turnstile)
    30          signal(turnstile)
    31
    32          /* critical point */
    33
    34          wait(mutex)
    35              count--
    36          signal(mutex)
    37
    38          if
    39          :: count == 0 ->
    40                          wait(turnstile)
```

```
41          :: else
42          fi
43
44          break    /* one only iteration */
45      od
46  }
47
48  init {
49      byte i
50
51      atomic {
52          for (i: 1 .. N) {
53              run P(i)
54          }
55      }
56      _nr_pr == 1 ->
57                      assert(turnstile == 0) /* must be locked */
58  }
```

Se dice que no es una solución correcta porque varios procesos pueden encontrar la condición en la línea 24 como cierta completando la línea 25. Lo mismo sucede con la condición en la línea 39 que puede poner en *deadlock* varios procesos.

El analizador proporciona los siguientes resultados:

```
$ spin -run lbs_3.7.1a.pml | expand
pan:1: invalid end state (at depth 46)
pan: wrote lbs_3.7.1a.pml.trail

(Spin Version 6.4.5 -- 1 January 2016)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +

State-vector 48 byte, depth reached 51, errors: 1
       52 states, stored
        3 states, matched
       55 transitions (= stored+matched)
       12 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.004       equivalent memory usage for states (stored*(State-vector + overhead))
    0.288       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  128.730       total actual memory usage



pan: elapsed time 0.01 seconds

$ spin -t lbs_3.7.1a.pml | expand
spin: trail ends after 47 steps
#processes: 2
                count = 0
```

```
                mutex = 1
                turnstile = 0
 47:    proc  1 (P:1) lbs_3.7.1a.pml:40 (state 23)
 47:    proc  0 (:init::1) lbs_3.7.1a.pml:56 (state 11)
4 processes created
```

según cuales se bloquea un solo proceso. No se necesita presentar ningún archivo, solamente se pide describir el escenario que lleva a la situación presentada.

Respuesta: Se puede suponer que 2 de 3 procesos P se bloquean en la línea 29, el 3er proceso encuentra `count` en 3 y envía una señal a `turnstile` (línea 25). Un proceso pasa a la línea 30 y libera el 2do proceso, el 2do proceso permite pasar al 3er proceso, y este último deja `turnstile` con el valor 1. Los tres procesos están en sus puntos críticos.

Según los resultados, 2 procesos terminan su código haciendo `break` pero el 3er proceso (junto con `init`) se quedan bloqueados. El 3er proceso se queda bloqueado en la línea 40 cuando `turnstile` tiene 0. Como el último proceso del grupo de 3 procesos, él hubiera que consumir la señal sobrante a `turnstile` que se quedó al llegar a su punto crítico del código. Pero esta señal extra desapareció.

La única explicación es que él no fue el único último sino había otro proceso que igualmente encontró la variable `count` en 0, y AMBOS procesos procedieron con la ejecución de la línea 40. Solamente uno pudo pasar usando la señal extra pendiente, el otro se quedó bloqueado.

Esta es la situación simétrica a la vista antes cuando todos los procesos encontraban al mismo tiempo que la variable `count` tiene el valor 3.

c) (`lbs_3.7.3b.pml`) (4 puntos – 36 min.) En el archivo **`lbs_3.7.3a.pml`**, está el modelo correspondiente al código de la sección 3.7.3:

```
$ cat -n lbs_3.7.3a.pml | expand
     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.7 Reusable barrier
     7      3.7.3 Reusable barrier non-solution #2
     8  */
     9
    10  #define N 3
    11
    12  #define wait(sem)   atomic { sem > 0; sem-- }
    13  #define signal(sem) sem++
    14
    15  byte count=0, mutex=1, turnstile=0 /* turnstile is locked */
    16
    17  proctype P(byte i) {
    18      do
    19      ::  wait(mutex)
    20              count++
    21              if
    22              :: count == N ->
    23                              signal(turnstile) /* the last opens turnstile */
    24              :: else
    25              fi
    26          signal(mutex)
```

```
    27
    28          wait(turnstile)
    29          signal(turnstile)
    30
    31          /* critical point */
    32
    33          wait(mutex)
    34              count--
    35              if
    36              :: count == 0 ->
    37                                  wait(turnstile) /* the last closes turnstile */
    38              :: else
    39              fi
    40          signal(mutex)
    41      od
    42  }
    43
    44  init {
    45      byte i
    46
    47      atomic {
    48          for (i: 1 .. N) {
    49              run P(i)
    50          }
    51      }
    52  }
```

Y el analizador proporciona los siguientes resultados:

```
$ spin -run lbs_3.7.3a.pml | expand
pan:1: invalid end state (at depth 5646)
pan: wrote lbs_3.7.3a.pml.trail

(Spin Version 6.4.5 -- 1 January 2016)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +

State-vector 48 byte, depth reached 5647, errors: 1
     5636 states, stored
        0 states, matched
     5636 transitions (= stored+matched)
       12 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.408       equivalent memory usage for states (stored*(State-vector + overhead))
    0.581       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  129.022       total actual memory usage



pan: elapsed time 0.01 seconds

$ spin -t -g lbs_3.7.3a.pml | expand
spin: lbs_3.7.3a.pml:23, Error: value (256->0 (8)) truncated in assignment
```

```
spin: lbs_3.7.3a.pml:23, Error: value (256->0 (8)) truncated in assignment
spin: trail ends after 5647 steps
#processes: 4
                count = 3
                mutex = 1
                turnstile = 0
5647:   proc  3 (P:1) lbs_3.7.3a.pml:28 (state 13)
5647:   proc  2 (P:1) lbs_3.7.3a.pml:28 (state 13)
5647:   proc  1 (P:1) lbs_3.7.3a.pml:28 (state 13)
5647:   proc  0 (:init::1) lbs_3.7.3a.pml:52 (state 11) <valid end state>
4 processes created
```

En la página 37 del libro se dice: “Tragically, the code is *still* not correct. Remember that this barrier will be inside a loop. So, after executing the last line each thread will go back to the rendezvous.”

En el archivo **`lbs_3.7.3b.pml`** prepare el modelo que exija una sincronización entre los procesos durante el paso de la barrera para que ningún proceso avance demasiado y que permita detectar el error antes posible.

Respuesta: Por el mensaje sobre el desbordamiento de la variable `turnstile` en la línea 23 parece que el 3er proceso fue el único que estaba haciendo sus iteraciones mientras que los otros procesos estaban esperando.

```
$ cat -n lbs_3.7.3b.pml | expand
     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.7 Reusable barrier
     7      3.7.3 Reusable barrier non-solution #2
     8  */
     9
    10  #define N 3
    11
    12  #define wait(sem)   atomic { sem > 0; sem-- }
    13  #define signal(sem) sem++
    14
    15  byte count=0, mutex=1, turnstile=0 /* turnstile is locked */
    16  byte loop[N+1]=1, sameloop=true
    17
    18  proctype P(byte i) {
    19      byte j
    20      do
    21      ::  wait(mutex)
    22              count++
    23              if
    24              :: count == N ->
    25                                  signal(turnstile) /* the last opens turnstile */
    26                                  for (j: 1 .. N-1) {
    27                                      sameloop = sameloop && (loop[j] == loop[j+1])
    28                                  }
    29                                  assert(sameloop)
    30              :: else
    31              fi
    32          signal(mutex)
    33
    34          wait(turnstile)
    35          signal(turnstile)
    36
    37          /* critical point */
```

```
38
39          wait(mutex)
40              count--
41              if
42              :: count == 0 ->
43                                  wait(turnstile) /* the last closes turnstile */
44              :: else
45              fi
46          signal(mutex)
47
48          if
49          :: loop[i] == 2 ->
50                                  break
51          :: else ->
52                          loop[i]++
53          fi
54      od
55  }
56
57  init {
58      byte i
59
60      atomic {
61          for (i: 1 .. N) {
62              run P(i)
63          }
64      }
65  }
```

```
$ spin -run lbs_3.7.3b.pml | expand
pan:1: assertion violated sameloop (at depth 57)
pan: wrote lbs_3.7.3b.pml.trail
...
State-vector 48 byte, depth reached 57, errors: 1
...
```

```
$ spin -t -p -g lbs_3.7.3b.pml | expand
...
 14:    proc  3 (P:1) lbs_3.7.3b.pml:21 (state 1)       [((mutex>0))]
 14:    proc  3 (P:1) lbs_3.7.3b.pml:21 (state 2)       [mutex = (mutex-1)]
                mutex = 0
 15:    proc  3 (P:1) lbs_3.7.3b.pml:22 (state 4)       [count = (count+1)]
                count = 1
 16:    proc  3 (P:1) lbs_3.7.3b.pml:30 (state 17)      [else]
 17:    proc  3 (P:1) lbs_3.7.3b.pml:32 (state 20)      [mutex = (mutex+1)]
                mutex = 1
 18:    proc  2 (P:1) lbs_3.7.3b.pml:21 (state 1)       [((mutex>0))]
 18:    proc  2 (P:1) lbs_3.7.3b.pml:21 (state 2)       [mutex = (mutex-1)]
                mutex = 0
 19:    proc  2 (P:1) lbs_3.7.3b.pml:22 (state 4)       [count = (count+1)]
                count = 2
 20:    proc  2 (P:1) lbs_3.7.3b.pml:30 (state 17)      [else]
 21:    proc  2 (P:1) lbs_3.7.3b.pml:32 (state 20)      [mutex = (mutex+1)]
                mutex = 1
 22:    proc  1 (P:1) lbs_3.7.3b.pml:21 (state 1)       [((mutex>0))]
 22:    proc  1 (P:1) lbs_3.7.3b.pml:21 (state 2)       [mutex = (mutex-1)]
                mutex = 0
 23:    proc  1 (P:1) lbs_3.7.3b.pml:22 (state 4)       [count = (count+1)]
                count = 3
 24:    proc  1 (P:1) lbs_3.7.3b.pml:24 (state 5)       [((count==3))]
 25:    proc  1 (P:1) lbs_3.7.3b.pml:25 (state 6)       [turnstile = (turnstile+1)]
                turnstile = 1
 26:    proc  1 (P:1) lbs_3.7.3b.pml:26 (state 7)       [j = 1]
```

```
 27:    proc  1 (P:1) lbs_3.7.3b.pml:26 (state 8)      [((j<=(3-1)))]
 28:    proc  3 (P:1) lbs_3.7.3b.pml:34 (state 21)     [((turnstile>0))]
 28:    proc  3 (P:1) lbs_3.7.3b.pml:34 (state 22)     [turnstile = (turnstile-1)]
                turnstile = 0
 29:    proc  3 (P:1) lbs_3.7.3b.pml:35 (state 24)     [turnstile = (turnstile+1)]
                turnstile = 1
 30:    proc  2 (P:1) lbs_3.7.3b.pml:34 (state 21)     [((turnstile>0))]
 30:    proc  2 (P:1) lbs_3.7.3b.pml:34 (state 22)     [turnstile = (turnstile-1)]
                turnstile = 0
 31:    proc  2 (P:1) lbs_3.7.3b.pml:35 (state 24)     [turnstile = (turnstile+1)]
                turnstile = 1
 32:    proc  1 (P:1) lbs_3.7.3b.pml:27 (state 9)      [sameloop =
                                                      (sameloop&&(loop[j]==loop[(j+1)]))]
 33:    proc  1 (P:1) lbs_3.7.3b.pml:26 (state 10)     [j = (j+1)]
 34:    proc  1 (P:1) lbs_3.7.3b.pml:26 (state 8)      [((j<=(3-1)))]
 35:    proc  1 (P:1) lbs_3.7.3b.pml:27 (state 9)      [sameloop =
                                                      (sameloop&&(loop[j]==loop[(j+1)]))]
 36:    proc  1 (P:1) lbs_3.7.3b.pml:26 (state 10)     [j = (j+1)]
 37:    proc  1 (P:1) lbs_3.7.3b.pml:28 (state 11)     [else]
 38:    proc  1 (P:1) lbs_3.7.3b.pml:29 (state 16)     [assert(sameloop)]
 39:    proc  1 (P:1) lbs_3.7.3b.pml:32 (state 20)     [mutex = (mutex+1)]
                mutex = 1
 40:    proc  3 (P:1) lbs_3.7.3b.pml:39 (state 25)     [((mutex>0))]
 40:    proc  3 (P:1) lbs_3.7.3b.pml:39 (state 26)     [mutex = (mutex-1)]
                mutex = 0
 41:    proc  3 (P:1) lbs_3.7.3b.pml:40 (state 28)     [count = (count-1)]
                count = 2
 42:    proc  3 (P:1) lbs_3.7.3b.pml:44 (state 33)     [else]
 43:    proc  3 (P:1) lbs_3.7.3b.pml:46 (state 36)     [mutex = (mutex+1)]
                mutex = 1
 44:    proc  3 (P:1) lbs_3.7.3b.pml:51 (state 39)     [else]
 45:    proc  3 (P:1) lbs_3.7.3b.pml:52 (state 40)     [loop[i] = (loop[i]+1)]
                loop[0] = 1
                loop[1] = 1
                loop[2] = 1
                loop[3] = 2
 46:    proc  3 (P:1) lbs_3.7.3b.pml:21 (state 1)      [((mutex>0))]
 46:    proc  3 (P:1) lbs_3.7.3b.pml:21 (state 2)      [mutex = (mutex-1)]
                mutex = 0
 47:    proc  3 (P:1) lbs_3.7.3b.pml:22 (state 4)      [count = (count+1)]
                count = 3
 48:    proc  3 (P:1) lbs_3.7.3b.pml:24 (state 5)      [((count==3))]
 49:    proc  3 (P:1) lbs_3.7.3b.pml:25 (state 6)      [turnstile = (turnstile+1)]
                turnstile = 2
 50:    proc  3 (P:1) lbs_3.7.3b.pml:26 (state 7)      [j = 1]
 51:    proc  3 (P:1) lbs_3.7.3b.pml:26 (state 8)      [((j<=(3-1)))]
 52:    proc  3 (P:1) lbs_3.7.3b.pml:27 (state 9)      [sameloop =
                                                      (sameloop&&(loop[j]==loop[(j+1)]))]
 53:    proc  3 (P:1) lbs_3.7.3b.pml:26 (state 10)     [j = (j+1)]
 54:    proc  3 (P:1) lbs_3.7.3b.pml:26 (state 8)      [((j<=(3-1)))]
 55:    proc  3 (P:1) lbs_3.7.3b.pml:27 (state 9)      [sameloop =
                                                      (sameloop&&(loop[j]==loop[(j+1)]))]
                sameloop = 0
 56:    proc  3 (P:1) lbs_3.7.3b.pml:26 (state 10)     [j = (j+1)]
 57:    proc  3 (P:1) lbs_3.7.3b.pml:28 (state 11)     [else]
spin: lbs_3.7.3b.pml:29, Error: assertion violated
spin: text of failed assertion: assert(sameloop)
 58:    proc  3 (P:1) lbs_3.7.3b.pml:29 (state 16)     [assert(sameloop)]
spin: trail ends after 58 steps
#processes: 4
                count = 3
                mutex = 0
                turnstile = 2
```

```
                loop[0] = 1
                loop[1] = 1
                loop[2] = 1
                loop[3] = 2
                sameloop = 0
 58:    proc  3 (P:1) lbs_3.7.3b.pml:32 (state 20)
 58:    proc  2 (P:1) lbs_3.7.3b.pml:39 (state 27)
 58:    proc  1 (P:1) lbs_3.7.3b.pml:34 (state 23)
 58:    proc  0 (:init::1) lbs_3.7.3b.pml:65 (state 11) <valid end state>
4 processes created
```

Pregunta 2. (**zeroB.pml**,**zeroC.pml**) (**4 puntos – 36 min.**) (*PCDP2E by M. Ben-Ari, Chapter 2, Exercise 5* (*Apt and Olderog*).) Usted ya conoce el modelo para el **Algorithm 2.11: Zero A**

Algorithm 2.11: Zero A	
boolean found	
p	**q**
integer i ← 0	integer j ← 1
p1: found ← false	q1: found ← false
p2: while not found	q2: while not found
p3: i ← i + 1	q3: j ← j − 1
p4: found ← f(i) = 0	q4: found ← f(j) = 0

```
$ cat -n zeroA.pml | expand
     1  #define MAX  100    /* 0..49, 50..99 */
     2  #define HALF MAX/2
     3
     4  #define f(x) (54 - x)
     5
     6  bool found
     7
     8  active proctype P() {
     9    byte i=HALF
    10
    11    found=false
    12    do
    13    :: found ->
    14                break
    15    :: else ->
    16                found = (f(i) == 0)
    17                if
    18                :: i==MAX-1 ->
    19                                i=HALF
    20                :: else     ->
    21                                i++
    22                fi
    23    od
    24  }
    25
    26  active proctype Q() {
    27    byte j = HALF-1
    28
    29    found = false
    30    do
    31    :: found ->
```

```
32                  break
33      :: else ->
34                  found = (f(j) == 0)
35                  if
36                  :: j==0 ->
37                             j=HALF-1
38                  :: else ->
39                             j--
40                  fi
41      od
42  }
```

Prepare en los archivos **zeroB.pml**, **zeroC.pml** y verifique los modelos correspondientes a los siguientes algoritmos:

Algorithm 2.12: Zero B	
boolean found ← false	
p	**q**
integer i ← 0	integer j ← 1
p1: while not found	q1: while not found
p2: i ← i + 1	q2: j ← j − 1
p3: found ← f(i) = 0	q3: found ← f(j) = 0

Algorithm 2.13: Zero C	
boolean found ← false	
p	**q**
integer i ← 0	integer j ← 1
p1: while not found	q1: while not found
p2: i ← i + 1	q2: j ← j − 1
p3: if f(i) = 0	q3: if f(j) = 0
p4: found ← true	q4: found ← true

Respuesta: (zeroB.pml) Es el modelo del material de preparación (zeroA_2.pml) ligeramente modificado:

```
$ cat -n zeroB.pml | expand
     1  #define MAX  4       /* 1..2, 3..4 */
     2  #define HALF MAX/2
     3
     4  #define f(x) (3 - x) /* P will find it */
     5
     6  #define ok (P@Pexited && Q@Qexited)
     7
     8  ltl { <>ok }
     9
    10  bool found=false
    11
    12  active proctype P() {
```

```
    13    byte i=HALF
    14
    15    do
    16    :: found ->
    17                break
    18    :: else ->
    19                i++
    20                if
    21                :: i==MAX+1 ->
    22                                i=HALF+1
    23                :: else
    24                fi
    25                found = (f(i) == 0)
    26    od
    27  Pexited:
    28  }
    29
    30  active proctype Q() {
    31    byte j = HALF+1
    32
    33    do
    34    :: found ->
    35                break
    36    :: else ->
    37                j--
    38                if
    39                :: j==0 ->
    40                            j=HALF
    41                :: else
    42                fi
    43                found = (f(j) == 0)
    44    od
    45  Qexited:
    46  }
```

```
$ spin -run -a -f zeroB.pml | expand
pan:1: acceptance cycle (at depth 124)
pan: wrote zeroB.pml.trail

(Spin Version 6.4.5 -- 1 January 2016)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             + (ltl_0)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness enabled)
        invalid end states      - (disabled by never claim)

State-vector 36 byte, depth reached 125, errors: 1
       59 states, stored (64 visited)
        9 states, matched
       73 transitions (= visited+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.004       equivalent memory usage for states (stored*(State-vector + overhead))
    0.289       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  128.730       total actual memory usage
```

```
pan: elapsed time 0 seconds
ltl ltl_0: <> (((P@Pexited)) && ((Q@Qexited)))

$ spin -t -p -g -l zeroB.pml | expand
ltl ltl_0: <> (((P@Pexited)) && ((Q@Qexited)))
starting claim 2
using statement merging
Never claim moves to line 4     [(!(((P._p==Pexited)&&(Q._p==Qexited))))]
  2:    proc  1 (Q:1) zeroB.pml:36 (state 3)    [else]
  4:    proc  1 (Q:1) zeroB.pml:37 (state 4)    [j = (j-1)]
                Q(1):j = 2
  6:    proc  1 (Q:1) zeroB.pml:41 (state 7)    [else]
  8:    proc  1 (Q:1) zeroB.pml:43 (state 10)   [found = ((3-j)==0)]
 10:    proc  1 (Q:1) zeroB.pml:36 (state 3)    [else]
 12:    proc  1 (Q:1) zeroB.pml:37 (state 4)    [j = (j-1)]
                Q(1):j = 1
 14:    proc  1 (Q:1) zeroB.pml:41 (state 7)    [else]
 16:    proc  1 (Q:1) zeroB.pml:43 (state 10)   [found = ((3-j)==0)]
 18:    proc  1 (Q:1) zeroB.pml:36 (state 3)    [else]
 20:    proc  1 (Q:1) zeroB.pml:37 (state 4)    [j = (j-1)]
                Q(1):j = 0
 22:    proc  0 (P:1) zeroB.pml:18 (state 3)    [else]
 24:    proc  1 (Q:1) zeroB.pml:39 (state 5)    [((j==0))]
 24:    proc  1 (Q:1) zeroB.pml:40 (state 6)    [j = (4/2)]
                Q(1):j = 2
 26:    proc  0 (P:1) zeroB.pml:19 (state 4)    [i = (i+1)]
                P(0):i = 3
 28:    proc  0 (P:1) zeroB.pml:23 (state 7)    [else]
 30:    proc  1 (Q:1) zeroB.pml:43 (state 10)   [found = ((3-j)==0)]
 32:    proc  1 (Q:1) zeroB.pml:36 (state 3)    [else]
 34:    proc  1 (Q:1) zeroB.pml:37 (state 4)    [j = (j-1)]
                Q(1):j = 1
 36:    proc  1 (Q:1) zeroB.pml:41 (state 7)    [else]
 38:    proc  1 (Q:1) zeroB.pml:43 (state 10)   [found = ((3-j)==0)]
 40:    proc  1 (Q:1) zeroB.pml:36 (state 3)    [else]
 42:    proc  1 (Q:1) zeroB.pml:37 (state 4)    [j = (j-1)]
                Q(1):j = 0
 44:    proc  1 (Q:1) zeroB.pml:39 (state 5)    [((j==0))]
 44:    proc  1 (Q:1) zeroB.pml:40 (state 6)    [j = (4/2)]
                Q(1):j = 2
 46:    proc  0 (P:1) zeroB.pml:25 (state 10)   [found = ((3-i)==0)]
                found = 1
 48:    proc  1 (Q:1) zeroB.pml:43 (state 10)   [found = ((3-j)==0)]
                found = 0
 50:    proc  1 (Q:1) zeroB.pml:36 (state 3)    [else]
 52:    proc  1 (Q:1) zeroB.pml:37 (state 4)    [j = (j-1)]
                Q(1):j = 1
 54:    proc  1 (Q:1) zeroB.pml:41 (state 7)    [else]
 56:    proc  1 (Q:1) zeroB.pml:43 (state 10)   [found = ((3-j)==0)]
 58:    proc  1 (Q:1) zeroB.pml:36 (state 3)    [else]
 60:    proc  1 (Q:1) zeroB.pml:37 (state 4)    [j = (j-1)]
                Q(1):j = 0
 62:    proc  1 (Q:1) zeroB.pml:39 (state 5)    [((j==0))]
 62:    proc  1 (Q:1) zeroB.pml:40 (state 6)    [j = (4/2)]
                Q(1):j = 2
 64:    proc  0 (P:1) zeroB.pml:18 (state 3)    [else]
 66:    proc  0 (P:1) zeroB.pml:19 (state 4)    [i = (i+1)]
                P(0):i = 4
 68:    proc  0 (P:1) zeroB.pml:23 (state 7)    [else]
 70:    proc  1 (Q:1) zeroB.pml:43 (state 10)   [found = ((3-j)==0)]
 72:    proc  1 (Q:1) zeroB.pml:36 (state 3)    [else]
```

```
 74:    proc  1 (Q:1) zeroB.pml:37 (state 4)    [j = (j-1)]
                Q(1):j = 1
 76:    proc  1 (Q:1) zeroB.pml:41 (state 7)    [else]
 78:    proc  1 (Q:1) zeroB.pml:43 (state 10)   [found = ((3-j)==0)]
 80:    proc  1 (Q:1) zeroB.pml:36 (state 3)    [else]
 82:    proc  1 (Q:1) zeroB.pml:37 (state 4)    [j = (j-1)]
                Q(1):j = 0
 84:    proc  1 (Q:1) zeroB.pml:39 (state 5)    [((j==0))]
 84:    proc  1 (Q:1) zeroB.pml:40 (state 6)    [j = (4/2)]
                Q(1):j = 2
 86:    proc  0 (P:1) zeroB.pml:25 (state 10)   [found = ((3-i)==0)]
 88:    proc  1 (Q:1) zeroB.pml:43 (state 10)   [found = ((3-j)==0)]
 90:    proc  1 (Q:1) zeroB.pml:36 (state 3)    [else]
 92:    proc  1 (Q:1) zeroB.pml:37 (state 4)    [j = (j-1)]
                Q(1):j = 1
 94:    proc  1 (Q:1) zeroB.pml:41 (state 7)    [else]
 96:    proc  1 (Q:1) zeroB.pml:43 (state 10)   [found = ((3-j)==0)]
 98:    proc  1 (Q:1) zeroB.pml:36 (state 3)    [else]
100:    proc  1 (Q:1) zeroB.pml:37 (state 4)    [j = (j-1)]
                Q(1):j = 0
102:    proc  1 (Q:1) zeroB.pml:39 (state 5)    [((j==0))]
102:    proc  1 (Q:1) zeroB.pml:40 (state 6)    [j = (4/2)]
                Q(1):j = 2
104:    proc  0 (P:1) zeroB.pml:18 (state 3)    [else]
106:    proc  0 (P:1) zeroB.pml:19 (state 4)    [i = (i+1)]
                P(0):i = 5
108:    proc  1 (Q:1) zeroB.pml:43 (state 10)   [found = ((3-j)==0)]
110:    proc  0 (P:1) zeroB.pml:21 (state 5)    [((i==(4+1)))]
110:    proc  0 (P:1) zeroB.pml:22 (state 6)    [i = ((4/2)+1)]
                P(0):i = 3
112:    proc  0 (P:1) zeroB.pml:25 (state 10)   [found = ((3-i)==0)]
                found = 1
114:    proc  1 (Q:1) zeroB.pml:34 (state 1)    [(found)]
116:    proc  1 (Q:1) zeroB.pml:46 (state 14)   [(1)]
118: proc 1 terminates
120:    proc  0 (P:1) zeroB.pml:16 (state 1)    [(found)]
122:    proc  0 (P:1) zeroB.pml:28 (state 14)   [(1)]
124: proc 0 terminates
  <<<<<START OF CYCLE>>>>>
spin: trail ends after 125 steps
#processes: 0
                found = 1
                Pexited = 0
                Qexited = 0
125:    proc  - (ltl_0:1) _spin_nvr.tmp:3 (state 3)
2 processes created
```

(zeroC.pml) En este modelo la variable found no se malogra:

```
$ cat -n zeroC.pml | expand
     1  #define MAX  4       /* 1..2, 3..4 */
     2  #define HALF MAX/2
     3
     4  #define f(x) (3 - x) /* P will find it */
     5
     6  #define ok (P@Pexited && Q@Qexited)
     7
     8  ltl { <>ok }
     9
    10  bool found=false
```

```
    11
    12  active proctype P() {
    13    byte i=HALF
    14
    15    do
    16    :: found ->
    17                break
    18    :: else ->
    19                i++
    20                if
    21                :: i==MAX+1 ->
    22                                i=HALF+1
    23                :: else
    24                fi
    25                if
    26                :: f(i)==0 ->
    27                                found=true
    28                :: else
    29                fi
    30    od
    31  Pexited:
    32  }
    33
    34  active proctype Q() {
    35    byte j = HALF+1
    36
    37    do
    38    :: found ->
    39                break
    40    :: else ->
    41                j--
    42                if
    43                :: j==0 ->
    44                              j=HALF
    45                :: else
    46                fi
    47                if
    48                :: f(j)==0 ->
    49                                found=true
    50                :: else
    51                fi
    52    od
    53  Qexited:
    54  }
```

```
$ spin -run -a -f zeroC.pml | expand
pan:1: acceptance cycle (at depth 62)
pan: wrote zeroC.pml.trail

(Spin Version 6.4.5 -- 1 January 2016)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             + (ltl_0)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness enabled)
        invalid end states      - (disabled by never claim)

State-vector 36 byte, depth reached 63, errors: 1
       31 states, stored (33 visited)
        4 states, matched
       37 transitions (= visited+matched)
```

```
       0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.002       equivalent memory usage for states (stored*(State-vector + overhead))
    0.288       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  128.730       total actual memory usage



pan: elapsed time 0 seconds
ltl ltl_0: <> (((P@Pexited)) && ((Q@Qexited)))

$ spin -t -p -g -l zeroC.pml | expand
ltl ltl_0: <> (((P@Pexited)) && ((Q@Qexited)))
starting claim 2
using statement merging
Never claim moves to line 4     [(!(((P._p==Pexited)&&(Q._p==Qexited))))]
  2:    proc  1 (Q:1) zeroC.pml:40 (state 3)    [else]
  4:    proc  1 (Q:1) zeroC.pml:41 (state 4)    [j = (j-1)]
                Q(1):j = 2
  6:    proc  1 (Q:1) zeroC.pml:45 (state 7)    [else]
  8:    proc  1 (Q:1) zeroC.pml:50 (state 12)   [else]
 10:    proc  1 (Q:1) zeroC.pml:40 (state 3)    [else]
 12:    proc  1 (Q:1) zeroC.pml:41 (state 4)    [j = (j-1)]
                Q(1):j = 1
 14:    proc  1 (Q:1) zeroC.pml:45 (state 7)    [else]
 16:    proc  1 (Q:1) zeroC.pml:50 (state 12)   [else]
 18:    proc  1 (Q:1) zeroC.pml:40 (state 3)    [else]
 20:    proc  1 (Q:1) zeroC.pml:41 (state 4)    [j = (j-1)]
                Q(1):j = 0
 22:    proc  0 (P:1) zeroC.pml:18 (state 3)    [else]
 24:    proc  1 (Q:1) zeroC.pml:43 (state 5)    [((j==0))]
 24:    proc  1 (Q:1) zeroC.pml:44 (state 6)    [j = (4/2)]
                Q(1):j = 2
 26:    proc  1 (Q:1) zeroC.pml:50 (state 12)   [else]
 28:    proc  0 (P:1) zeroC.pml:19 (state 4)    [i = (i+1)]
                P(0):i = 3
 30:    proc  0 (P:1) zeroC.pml:23 (state 7)    [else]
 32:    proc  0 (P:1) zeroC.pml:26 (state 10)   [(((3-i)==0))]
 34:    proc  1 (Q:1) zeroC.pml:40 (state 3)    [else]
 36:    proc  1 (Q:1) zeroC.pml:41 (state 4)    [j = (j-1)]
                Q(1):j = 1
 38:    proc  1 (Q:1) zeroC.pml:45 (state 7)    [else]
 40:    proc  1 (Q:1) zeroC.pml:50 (state 12)   [else]
 42:    proc  1 (Q:1) zeroC.pml:40 (state 3)    [else]
 44:    proc  1 (Q:1) zeroC.pml:41 (state 4)    [j = (j-1)]
                Q(1):j = 0
 46:    proc  1 (Q:1) zeroC.pml:43 (state 5)    [((j==0))]
 46:    proc  1 (Q:1) zeroC.pml:44 (state 6)    [j = (4/2)]
                Q(1):j = 2
 48:    proc  1 (Q:1) zeroC.pml:50 (state 12)   [else]
 50:    proc  0 (P:1) zeroC.pml:27 (state 11)   [found = 1]
                found = 1
 52:    proc  1 (Q:1) zeroC.pml:38 (state 1)    [(found)]
 54:    proc  1 (Q:1) zeroC.pml:54 (state 18)   [(1)]
 56: proc 1 terminates
 58:    proc  0 (P:1) zeroC.pml:16 (state 1)    [(found)]
 60:    proc  0 (P:1) zeroC.pml:32 (state 18)   [(1)]
 62: proc 0 terminates
  <<<<<START OF CYCLE>>>>>
```

```
spin: trail ends after 63 steps
#processes: 0
                found = 1
                Pexited = 0
                Qexited = 0
 63:    proc  - (ltl_0:1) _spin_nvr.tmp:3 (state 3)
2 processes created
```

Pregunta 3. (`alg2.16a.pml`, `alg2.16b.pml`, `alg2.16c.pml`) (6 puntos – 54 min.) (*PCDP2E by M. Ben-Ari, Chapter 2, Exercise 5* (*Apt and Olderog*).) Consider the following algorithm where each of ten processes executes the statements with *i* set to a different number in 1, …, 10:

Algorithm 2.16: Concurrent algorithm A
integer array [1..10] C ← ten *distinct* initial values integer array [1..10] D
integer myNumber, count p1: myNumber ← C[i] p2: count ← number of elements of C less than myNumber p3: D[count + 1] ← myNumber

a) (`alg2.16a.pml`) (2 puntos – 18 min.) En el archivo **`alg2.16a.pml`** prepare el modelo para el algoritmo dado.

Respuesta:

```
$ cat -n alg2.16a.pml | expand
     1  #define N 5
     2
     3  int C[N+1],D[N+1]   /* index 0 is not used here! */
     4
     5  proctype P(byte i) {
     6      int myNumber=C[i], count=0, j
     7
     8      for (j: 1 .. N) {
     9          if
    10          :: C[j] < myNumber -> count++
    11          :: else
    12          fi
    13      }
    14      D[count+1] = myNumber
    15  }
    16
    17
    18  init {
    19      byte i
    20
    21      C[1]=41; C[2]=13; C[3]=7; C[4]=57; C[5]=51
    22      atomic {
    23          for (i: 1 .. N) {
    24              run P(i)
    25          }
    26      }
    27      _nr_pr==1
    28      for (i: 1 .. N) {
    29          printf("%d ",D[i])
```

```
    30      }
    31      printf("\n")
    32  }
```

```
$ spin alg2.16a.pml
      7       13       41       51       57
6 processes created
```

b) (`alg2.16b.pml`) (**2 puntos – 18 min.**) If *D* in line *p3* is replaced by *C*, sometimes may occur the indexing error. Why? What scenario with?

```
$ cat -n alg2.16b.pml | expand
     1  #define N 5
     2
     3  int C[N+1],D[N+1]   /* index 0 is not used here! */
     4
     5  proctype P(byte i) {
     6      int myNumber=C[i], count=0, j
     7
     8      for (j: 1 .. N) {
     9          if
    10          :: C[j] < myNumber -> count++
    11          :: else
    12          fi
    13      }
    14      C[count+1] = myNumber
    15  }
    16
    17
    18  init {
    19      byte i
    20
    21      C[1]=41; C[2]=13; C[3]=7; C[4]=57; C[5]=51
    22      atomic {
    23          for (i: 1 .. N) {
    24              run P(i)
    25          }
    26      }
    27      _nr_pr==1
    28      for (i: 1 .. N) {
    29          printf("%d ",C[i])
    30      }
    31      printf("\n")
    32  }
```

```
$ spin alg2.16b.pml
      7       13       41       51       57
6 processes created

$ spin alg2.16b.pml
      7       13       41       51       57
6 processes created

$ spin alg2.16b.pml
      7       13       41       51       57
6 processes created

$ spin alg2.16b.pml
spin: indexing C[6] - size is 6
spin: alg2.16b.pml:14, Error: indexing array 'C'
      7       13       41       51       51
6 processes created
```

El proceso P(4) (para ubicar la posición para el número 57) trabajó después del proceso P(5) que ya ubicó la posición correcta para su número 51. Y el proceso P(4) encuentra el número 51 (menor que 57) 2 veces, lo que lleva al cálculo de la posición 5+1 = 6, la posición que está fuera del rango del índice.

Podremos añadir el aserto para exigir que los elementos del arreglo siempre sean distintos (`alg2.16b_1.pml`):

```
$ cat -n alg2.16b_1.pml | expand
     1  #define N 5
     2
     3  int C[N+1],D[N+1]   /* index 0 is not used here! */
     4
     5  proctype P(byte i) {
     6      int myNumber=C[i], count=0, j=1
     7      int k=1, eq=0
     8      bool dstnct=true
     9
    10      for (j: 1 .. N) {
    11          if
    12          :: C[j] < myNumber -> count++
    13          :: else
    14          fi
    15      }
    16
    17      for (j: 1 .. N) {
    18          eq=0
    19          for (k: 1 .. N) {
    20              if
    21              :: C[j]==C[k] -> eq++
    22              :: else
    23              fi
    24          }
    25          dstnct=dstnct && (eq==1)
    26      }
    27      assert(dstnct)
    28      C[count+1] = myNumber
    29  }
    30
    31
    32  init {
    33      byte i
    34
    35      C[1]=41; C[2]=13; C[3]=7; C[4]=57; C[5]=51
    36      atomic {
    37          for (i: 1 .. N) {
    38              run P(i)
    39          }
    40      }
    41      _nr_pr==1
    42      for (i: 1 .. N) {
    43          printf("%d ",C[i])
    44      }
    45      printf("\n")
    46  }

$ spin -run alg2.16b_1.pml | expand
pan:1: assertion violated dstnct (at depth 251)
pan: wrote alg2.16b_1.pml.trail
```

```
(Spin Version 6.4.5 -- 1 January 2016)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +

State-vector 160 byte, depth reached 251, errors: 1
      234 states, stored
        0 states, matched
      234 transitions (= stored+matched)
       18 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.042       equivalent memory usage for states (stored*(State-vector + overhead))
    0.286       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  128.730       total actual memory usage


pan: elapsed time 0 seconds

$ spin -t -p -g -l alg2.16b_1.pml
using statement merging
  1:   proc  0 (:init::1) alg2.16b_1.pml:35 (state 1)      [C[1] = 41]
               C[0] = 0
               C[1] = 41
               C[2] = 0
               C[3] = 0
               C[4] = 0
               C[5] = 0
  2:   proc  0 (:init::1) alg2.16b_1.pml:35 (state 2)      [C[2] = 13]
               C[0] = 0
               C[1] = 41
               C[2] = 13
               C[3] = 0
               C[4] = 0
               C[5] = 0
  3:   proc  0 (:init::1) alg2.16b_1.pml:35 (state 3)      [C[3] = 7]
               C[0] = 0
               C[1] = 41
               C[2] = 13
               C[3] = 7
               C[4] = 0
               C[5] = 0
  4:   proc  0 (:init::1) alg2.16b_1.pml:35 (state 4)      [C[4] = 57]
               C[0] = 0
               C[1] = 41
               C[2] = 13
               C[3] = 7
               C[4] = 57
               C[5] = 0
  5:   proc  0 (:init::1) alg2.16b_1.pml:35 (state 5)      [C[5] = 51]
               C[0] = 0
               C[1] = 41
               C[2] = 13
               C[3] = 7
               C[4] = 57
```

```
            C[5] = 51
  6:   proc  0 (:init::1) alg2.16b_1.pml:37 (state 6)      [i = 1]
            :init:(0):i = 1
  7:   proc  0 (:init::1) alg2.16b_1.pml:37 (state 7)      [((i<=5))]
Starting P with pid 1
  8:   proc  0 (:init::1) alg2.16b_1.pml:38 (state 8)      [(run P(i))]
  9:   proc  0 (:init::1) alg2.16b_1.pml:37 (state 9)      [i = (i+1)]
            :init:(0):i = 2
 10:   proc  0 (:init::1) alg2.16b_1.pml:37 (state 7)      [((i<=5))]
Starting P with pid 2
 11:   proc  0 (:init::1) alg2.16b_1.pml:38 (state 8)      [(run P(i))]
 12:   proc  0 (:init::1) alg2.16b_1.pml:37 (state 9)      [i = (i+1)]
            :init:(0):i = 3
 13:   proc  0 (:init::1) alg2.16b_1.pml:37 (state 7)      [((i<=5))]
Starting P with pid 3
 14:   proc  0 (:init::1) alg2.16b_1.pml:38 (state 8)      [(run P(i))]
 15:   proc  0 (:init::1) alg2.16b_1.pml:37 (state 9)      [i = (i+1)]
            :init:(0):i = 4
 16:   proc  0 (:init::1) alg2.16b_1.pml:37 (state 7)      [((i<=5))]
Starting P with pid 4
 17:   proc  0 (:init::1) alg2.16b_1.pml:38 (state 8)      [(run P(i))]
 18:   proc  0 (:init::1) alg2.16b_1.pml:37 (state 9)      [i = (i+1)]
            :init:(0):i = 5
 19:   proc  0 (:init::1) alg2.16b_1.pml:37 (state 7)      [((i<=5))]
Starting P with pid 5
 20:   proc  0 (:init::1) alg2.16b_1.pml:38 (state 8)      [(run P(i))]
 21:   proc  0 (:init::1) alg2.16b_1.pml:37 (state 9)      [i = (i+1)]
            :init:(0):i = 6
 22:   proc  0 (:init::1) alg2.16b_1.pml:39 (state 10)     [else]
 23:   proc  0 (:init::1) alg2.16b_1.pml:39 (state 11)     [goto :b3]
 24:   proc  0 (:init::1) alg2.16b_1.pml:39 (state 14)     [break]
 25:   proc  5 (P:1) alg2.16b_1.pml:10 (state 1)     [j = 1]
            P(5):j = 1
 26:   proc  5 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
 27:   proc  4 (P:1) alg2.16b_1.pml:10 (state 1)     [j = 1]
            P(4):j = 1
 28:   proc  4 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
 29:   proc  3 (P:1) alg2.16b_1.pml:10 (state 1)     [j = 1]
            P(3):j = 1
 30:   proc  3 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
 31:   proc  2 (P:1) alg2.16b_1.pml:10 (state 1)     [j = 1]
            P(2):j = 1
 32:   proc  2 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
 33:   proc  1 (P:1) alg2.16b_1.pml:10 (state 1)     [j = 1]
            P(1):j = 1
 34:   proc  1 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
 35:   proc  5 (P:1) alg2.16b_1.pml:12 (state 3)     [((C[j]<myNumber))]
 36:   proc  5 (P:1) alg2.16b_1.pml:12 (state 4)     [count = (count+1)]
            P(5):count = 1
 36:   proc  5 (P:1) alg2.16b_1.pml:10 (state 8)     [j = (j+1)]
            P(5):j = 2
            P(5):count = 1
 37:   proc  5 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
 38:   proc  5 (P:1) alg2.16b_1.pml:12 (state 3)     [((C[j]<myNumber))]
 39:   proc  5 (P:1) alg2.16b_1.pml:12 (state 4)     [count = (count+1)]
            P(5):count = 2
 39:   proc  5 (P:1) alg2.16b_1.pml:10 (state 8)     [j = (j+1)]
            P(5):j = 3
            P(5):count = 2
 40:   proc  5 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
 41:   proc  5 (P:1) alg2.16b_1.pml:12 (state 3)     [((C[j]<myNumber))]
 42:   proc  5 (P:1) alg2.16b_1.pml:12 (state 4)     [count = (count+1)]
            P(5):count = 3
```

```
42:   proc  5 (P:1) alg2.16b_1.pml:10 (state 8)     [j = (j+1)]
            P(5):j = 4
            P(5):count = 3
43:   proc  5 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
44:   proc  5 (P:1) alg2.16b_1.pml:13 (state 5)     [else]
45:   proc  5 (P:1) alg2.16b_1.pml:10 (state 8)     [j = (j+1)]
            P(5):j = 5
46:   proc  5 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
47:   proc  5 (P:1) alg2.16b_1.pml:13 (state 5)     [else]
48:   proc  5 (P:1) alg2.16b_1.pml:10 (state 8)     [j = (j+1)]
            P(5):j = 6
49:   proc  5 (P:1) alg2.16b_1.pml:15 (state 9)     [else]
50:   proc  5 (P:1) alg2.16b_1.pml:17 (state 14)    [j = 1]
            P(5):j = 1
51:   proc  5 (P:1) alg2.16b_1.pml:17 (state 15)    [((j<=5))]
51:   proc  5 (P:1) alg2.16b_1.pml:18 (state 16)    [eq = 0]
            P(5):eq = 0
51:   proc  5 (P:1) alg2.16b_1.pml:19 (state 17)    [k = 1]
            P(5):eq = 0
            P(5):k = 1
52:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
53:   proc  5 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
54:   proc  5 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
            P(5):eq = 1
54:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(5):eq = 1
            P(5):k = 2
55:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
56:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
57:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(5):k = 3
58:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
59:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
60:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(5):k = 4
61:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
62:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
63:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(5):k = 5
64:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
65:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
66:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(5):k = 6
67:   proc  5 (P:1) alg2.16b_1.pml:24 (state 25)    [else]
68:   proc  5 (P:1) alg2.16b_1.pml:25 (state 30)    [dstnct = (dstnct&&(eq==1))]
            P(5):dstnct = 1
68:   proc  5 (P:1) alg2.16b_1.pml:17 (state 31)    [j = (j+1)]
            P(5):dstnct = 1
            P(5):j = 2
69:   proc  5 (P:1) alg2.16b_1.pml:17 (state 15)    [((j<=5))]
69:   proc  5 (P:1) alg2.16b_1.pml:18 (state 16)    [eq = 0]
            P(5):eq = 0
69:   proc  5 (P:1) alg2.16b_1.pml:19 (state 17)    [k = 1]
            P(5):eq = 0
            P(5):k = 1
70:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
71:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
72:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(5):k = 2
73:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
74:   proc  5 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
75:   proc  5 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
            P(5):eq = 1
```

```
 75:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):eq = 1
             P(5):k = 3
 76:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
 77:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
 78:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 4
 79:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
 80:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
 81:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 5
 82:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
 83:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
 84:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 6
 85:   proc  5 (P:1) alg2.16b_1.pml:24 (state 25)    [else]
 86:   proc  5 (P:1) alg2.16b_1.pml:25 (state 30)    [dstnct = (dstnct&&(eq==1))]
             P(5):dstnct = 1
 86:   proc  5 (P:1) alg2.16b_1.pml:17 (state 31)    [j = (j+1)]
             P(5):dstnct = 1
             P(5):j = 3
 87:   proc  5 (P:1) alg2.16b_1.pml:17 (state 15)    [((j<=5))]
 87:   proc  5 (P:1) alg2.16b_1.pml:18 (state 16)    [eq = 0]
             P(5):eq = 0
 87:   proc  5 (P:1) alg2.16b_1.pml:19 (state 17)    [k = 1]
             P(5):eq = 0
             P(5):k = 1
 88:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
 89:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
 90:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 2
 91:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
 92:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
 93:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 3
 94:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
 95:   proc  5 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
 96:   proc  5 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
             P(5):eq = 1
 96:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):eq = 1
             P(5):k = 4
 97:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
 98:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
 99:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 5
100:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
101:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
102:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 6
103:   proc  5 (P:1) alg2.16b_1.pml:24 (state 25)    [else]
104:   proc  5 (P:1) alg2.16b_1.pml:25 (state 30)    [dstnct = (dstnct&&(eq==1))]
             P(5):dstnct = 1
104:   proc  5 (P:1) alg2.16b_1.pml:17 (state 31)    [j = (j+1)]
             P(5):dstnct = 1
             P(5):j = 4
105:   proc  5 (P:1) alg2.16b_1.pml:17 (state 15)    [((j<=5))]
105:   proc  5 (P:1) alg2.16b_1.pml:18 (state 16)    [eq = 0]
             P(5):eq = 0
105:   proc  5 (P:1) alg2.16b_1.pml:19 (state 17)    [k = 1]
             P(5):eq = 0
             P(5):k = 1
106:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
```

```
107:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
108:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 2
109:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
110:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
111:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 3
112:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
113:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
114:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 4
115:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
116:   proc  5 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
117:   proc  5 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
             P(5):eq = 1
117:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):eq = 1
             P(5):k = 5
118:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
119:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
120:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 6
121:   proc  5 (P:1) alg2.16b_1.pml:24 (state 25)    [else]
122:   proc  5 (P:1) alg2.16b_1.pml:25 (state 30)    [dstnct = (dstnct&&(eq==1))]
             P(5):dstnct = 1
122:   proc  5 (P:1) alg2.16b_1.pml:17 (state 31)    [j = (j+1)]
             P(5):dstnct = 1
             P(5):j = 5
123:   proc  5 (P:1) alg2.16b_1.pml:17 (state 15)    [((j<=5))]
123:   proc  5 (P:1) alg2.16b_1.pml:18 (state 16)    [eq = 0]
             P(5):eq = 0
123:   proc  5 (P:1) alg2.16b_1.pml:19 (state 17)    [k = 1]
             P(5):eq = 0
             P(5):k = 1
124:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
125:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
126:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 2
127:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
128:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
129:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 3
130:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
131:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
132:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 4
133:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
134:   proc  5 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
135:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):k = 5
136:   proc  5 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
137:   proc  5 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
138:   proc  5 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
             P(5):eq = 1
138:   proc  5 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(5):eq = 1
             P(5):k = 6
139:   proc  5 (P:1) alg2.16b_1.pml:24 (state 25)    [else]
140:   proc  5 (P:1) alg2.16b_1.pml:25 (state 30)    [dstnct = (dstnct&&(eq==1))]
             P(5):dstnct = 1
140:   proc  5 (P:1) alg2.16b_1.pml:17 (state 31)    [j = (j+1)]
             P(5):dstnct = 1
             P(5):j = 6
```

```
141:   proc  5 (P:1) alg2.16b_1.pml:26 (state 32)    [else]
142:   proc  5 (P:1) alg2.16b_1.pml:27 (state 37)    [assert(dstnct)]
143:   proc  5 (P:1) alg2.16b_1.pml:28 (state 38)    [C[(count+1)] = myNumber]
              C[0] = 0
              C[1] = 41
              C[2] = 13
              C[3] = 7
              C[4] = 51
              C[5] = 51
144: proc 5 terminates
145:   proc  4 (P:1) alg2.16b_1.pml:12 (state 3)     [((C[j]<myNumber))]
146:   proc  4 (P:1) alg2.16b_1.pml:12 (state 4)     [count = (count+1)]
              P(4):count = 1
146:   proc  4 (P:1) alg2.16b_1.pml:10 (state 8)     [j = (j+1)]
              P(4):j = 2
              P(4):count = 1
147:   proc  4 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
148:   proc  4 (P:1) alg2.16b_1.pml:12 (state 3)     [((C[j]<myNumber))]
149:   proc  4 (P:1) alg2.16b_1.pml:12 (state 4)     [count = (count+1)]
              P(4):count = 2
149:   proc  4 (P:1) alg2.16b_1.pml:10 (state 8)     [j = (j+1)]
              P(4):j = 3
              P(4):count = 2
150:   proc  4 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
151:   proc  4 (P:1) alg2.16b_1.pml:12 (state 3)     [((C[j]<myNumber))]
152:   proc  4 (P:1) alg2.16b_1.pml:12 (state 4)     [count = (count+1)]
              P(4):count = 3
152:   proc  4 (P:1) alg2.16b_1.pml:10 (state 8)     [j = (j+1)]
              P(4):j = 4
              P(4):count = 3
153:   proc  4 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
154:   proc  4 (P:1) alg2.16b_1.pml:12 (state 3)     [((C[j]<myNumber))]
155:   proc  4 (P:1) alg2.16b_1.pml:12 (state 4)     [count = (count+1)]
              P(4):count = 4
155:   proc  4 (P:1) alg2.16b_1.pml:10 (state 8)     [j = (j+1)]
              P(4):j = 5
              P(4):count = 4
156:   proc  4 (P:1) alg2.16b_1.pml:10 (state 2)     [((j<=5))]
157:   proc  4 (P:1) alg2.16b_1.pml:12 (state 3)     [((C[j]<myNumber))]
158:   proc  4 (P:1) alg2.16b_1.pml:12 (state 4)     [count = (count+1)]
              P(4):count = 5
158:   proc  4 (P:1) alg2.16b_1.pml:10 (state 8)     [j = (j+1)]
              P(4):j = 6
              P(4):count = 5
159:   proc  4 (P:1) alg2.16b_1.pml:15 (state 9)     [else]
160:   proc  4 (P:1) alg2.16b_1.pml:17 (state 14)    [j = 1]
              P(4):j = 1
161:   proc  4 (P:1) alg2.16b_1.pml:17 (state 15)    [((j<=5))]
161:   proc  4 (P:1) alg2.16b_1.pml:18 (state 16)    [eq = 0]
              P(4):eq = 0
161:   proc  4 (P:1) alg2.16b_1.pml:19 (state 17)    [k = 1]
              P(4):eq = 0
              P(4):k = 1
162:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
163:   proc  4 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
164:   proc  4 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
              P(4):eq = 1
164:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
              P(4):eq = 1
              P(4):k = 2
165:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
166:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
167:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
```

```
            P(4):k = 3
168:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
169:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
170:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 4
171:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
172:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
173:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 5
174:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
175:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
176:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 6
177:   proc  4 (P:1) alg2.16b_1.pml:24 (state 25)    [else]
178:   proc  4 (P:1) alg2.16b_1.pml:25 (state 30)    [dstnct = (dstnct&&(eq==1))]
            P(4):dstnct = 1
178:   proc  4 (P:1) alg2.16b_1.pml:17 (state 31)    [j = (j+1)]
            P(4):dstnct = 1
            P(4):j = 2
179:   proc  4 (P:1) alg2.16b_1.pml:17 (state 15)    [((j<=5))]
179:   proc  4 (P:1) alg2.16b_1.pml:18 (state 16)    [eq = 0]
            P(4):eq = 0
179:   proc  4 (P:1) alg2.16b_1.pml:19 (state 17)    [k = 1]
            P(4):eq = 0
            P(4):k = 1
180:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
181:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
182:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 2
183:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
184:   proc  4 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
185:   proc  4 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
            P(4):eq = 1
185:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):eq = 1
            P(4):k = 3
186:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
187:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
188:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 4
189:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
190:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
191:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 5
192:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
193:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
194:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 6
195:   proc  4 (P:1) alg2.16b_1.pml:24 (state 25)    [else]
196:   proc  4 (P:1) alg2.16b_1.pml:25 (state 30)    [dstnct = (dstnct&&(eq==1))]
            P(4):dstnct = 1
196:   proc  4 (P:1) alg2.16b_1.pml:17 (state 31)    [j = (j+1)]
            P(4):dstnct = 1
            P(4):j = 3
197:   proc  4 (P:1) alg2.16b_1.pml:17 (state 15)    [((j<=5))]
197:   proc  4 (P:1) alg2.16b_1.pml:18 (state 16)    [eq = 0]
            P(4):eq = 0
197:   proc  4 (P:1) alg2.16b_1.pml:19 (state 17)    [k = 1]
            P(4):eq = 0
            P(4):k = 1
198:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
199:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
200:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
```

```
            P(4):k = 2
201:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
202:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
203:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 3
204:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
205:   proc  4 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
206:   proc  4 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
            P(4):eq = 1
206:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):eq = 1
            P(4):k = 4
207:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
208:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
209:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 5
210:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
211:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
212:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 6
213:   proc  4 (P:1) alg2.16b_1.pml:24 (state 25)    [else]
214:   proc  4 (P:1) alg2.16b_1.pml:25 (state 30)    [dstnct = (dstnct&&(eq==1))]
            P(4):dstnct = 1
214:   proc  4 (P:1) alg2.16b_1.pml:17 (state 31)    [j = (j+1)]
            P(4):dstnct = 1
            P(4):j = 4
215:   proc  4 (P:1) alg2.16b_1.pml:17 (state 15)    [((j<=5))]
215:   proc  4 (P:1) alg2.16b_1.pml:18 (state 16)    [eq = 0]
            P(4):eq = 0
215:   proc  4 (P:1) alg2.16b_1.pml:19 (state 17)    [k = 1]
            P(4):eq = 0
            P(4):k = 1
216:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
217:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
218:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 2
219:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
220:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
221:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 3
222:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
223:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
224:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):k = 4
225:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
226:   proc  4 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
227:   proc  4 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
            P(4):eq = 1
227:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):eq = 1
            P(4):k = 5
228:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
229:   proc  4 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
230:   proc  4 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
            P(4):eq = 2
230:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
            P(4):eq = 2
            P(4):k = 6
231:   proc  4 (P:1) alg2.16b_1.pml:24 (state 25)    [else]
232:   proc  4 (P:1) alg2.16b_1.pml:25 (state 30)    [dstnct = (dstnct&&(eq==1))]
            P(4):dstnct = 0
232:   proc  4 (P:1) alg2.16b_1.pml:17 (state 31)    [j = (j+1)]
            P(4):dstnct = 0
```

```
             P(4):j = 5
233:   proc  4 (P:1) alg2.16b_1.pml:17 (state 15)    [((j<=5))]
233:   proc  4 (P:1) alg2.16b_1.pml:18 (state 16)    [eq = 0]
             P(4):eq = 0
233:   proc  4 (P:1) alg2.16b_1.pml:19 (state 17)    [k = 1]
             P(4):eq = 0
             P(4):k = 1
234:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
235:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
236:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(4):k = 2
237:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
238:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
239:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(4):k = 3
240:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
241:   proc  4 (P:1) alg2.16b_1.pml:22 (state 21)    [else]
242:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(4):k = 4
243:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
244:   proc  4 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
245:   proc  4 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
             P(4):eq = 1
245:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(4):eq = 1
             P(4):k = 5
246:   proc  4 (P:1) alg2.16b_1.pml:19 (state 18)    [((k<=5))]
247:   proc  4 (P:1) alg2.16b_1.pml:21 (state 19)    [((C[j]==C[k]))]
248:   proc  4 (P:1) alg2.16b_1.pml:21 (state 20)    [eq = (eq+1)]
             P(4):eq = 2
248:   proc  4 (P:1) alg2.16b_1.pml:19 (state 24)    [k = (k+1)]
             P(4):eq = 2
             P(4):k = 6
249:   proc  4 (P:1) alg2.16b_1.pml:24 (state 25)    [else]
250:   proc  4 (P:1) alg2.16b_1.pml:25 (state 30)    [dstnct = (dstnct&&(eq==1))]
             P(4):dstnct = 0
250:   proc  4 (P:1) alg2.16b_1.pml:17 (state 31)    [j = (j+1)]
             P(4):dstnct = 0
             P(4):j = 6
251:   proc  4 (P:1) alg2.16b_1.pml:26 (state 32)    [else]
spin: alg2.16b_1.pml:27, Error: assertion violated
spin: text of failed assertion: assert(dstnct)
252:   proc  4 (P:1) alg2.16b_1.pml:27 (state 37)    [assert(dstnct)]
spin: trail ends after 252 steps
#processes: 5
             C[0] = 0
             C[1] = 41
             C[2] = 13
             C[3] = 7
             C[4] = 51
             C[5] = 51
             D[0] = 0
             D[1] = 0
             D[2] = 0
             D[3] = 0
             D[4] = 0
             D[5] = 0
252:   proc  4 (P:1) alg2.16b_1.pml:28 (state 38)
252:   proc  3 (P:1) alg2.16b_1.pml:11 (state 6)
252:   proc  2 (P:1) alg2.16b_1.pml:11 (state 6)
252:   proc  1 (P:1) alg2.16b_1.pml:11 (state 6)
252:   proc  0 (:init::1) alg2.16b_1.pml:41 (state 16)
6 processes created
```

c) (`alg2.16c.pml`) (**2 puntos – 18 min.**) What would happen if the array *C* were initialized with values that are not all distinct? Correct the algorithm to take care of this case.

Respuesta: Por ejemplo, de la siguiente manera, contando la cantidad de valores iguales:

```
$ cat -n alg2.16c.pml | expand
     1  #define N 5
     2
     3  int C[N+1],D[N+1]   /* index 0 is not used here! */
     4
     5  proctype P(byte i) {
     6      int myNumber=C[i], count=0, j
     7      int equals=0
     8
     9      for (j: 1 .. N) {
    10          if
    11          :: C[j] < myNumber -> count++
    12          :: C[j] == myNumber -> equals++
    13          :: else
    14          fi
    15      }
    16      for (j: 1 .. equals) {
    17          D[count+j] = myNumber
    18      }
    19  }
    20
    21
    22  init {
    23      byte i
    24
    25      C[1]=7; C[2]=13; C[3]=7; C[4]=13; C[5]=13
    26      atomic {
    27          for (i: 1 .. N) {
    28              run P(i)
    29          }
    30      }
    31      _nr_pr==1
    32      for (i: 1 .. N) {
    33          printf("%d ",D[i])
    34      }
    35      printf("\n")
    36  }

$ spin alg2.16c.pml | expand
      7       7       13       13       13
6 processes created
```

Profesor: V. Khlebnikov
Pando, 14 de octubre de 2016