Assignment 1 (PROMELA) (10p)

(a) The PROMELA code below is an incomplete implementation of a threshold filter. The filter receives values on channel C and forwards them based on the threshold variables $t1$ and $t2$. When a received value is greater or equal than $t1$ it is sent on channel $O1$, if it is between $t2$ and $t1$ it is sent on channel $O2$, and if it is smaller than $t2$ it is dropped. For verification we use a listener process that receives values on channels $O1$ and $O2$ and *asserts* that these values satisfy the corresponding thresholds. Your task is to implement the filter and listener processes. You may assume that $t1 > t2$.

```
chan C  = [0] of {byte}
chan O1 = [0] of {byte}
chan O2 = [0] of {byte}

byte t1 = 4; /* thresholds */
byte t2 = 2;

active proctype generator() {
end:
  do
    :: C ! 0
    :: C ! 1
    :: C ! 2
    :: C ! 3
    :: C ! 4
    :: C ! 5
  od
}

active proctype filter ()  {  /* ... */  }

active proctype listener() {  /* ... */  }
```

(For part (b) of this assignment, see next page)

(b) The PROMELA model below has a flaw: it may deadlock. Explain why a deadlock is possible and show a trail of channel messages that exhibits it.

```
mtype {msgA, msgB};
chan C1 = [1] of {mtype}; /* buffered channel    */
chan C2 = [0] of {mtype}; /* synchronous channel */

active proctype P() {
  C1 ! msgA
}

active proctype Q() {
  C2 ! msgB
}

active proctype Z() {
  byte x;
end:
  do
    :: C1 ?  x ->
         if  :: C1 ! x
             :: C2 ! x
         fi
    :: C2 ?  x ->
         if  :: C1 ! x
             :: C2 ! x
         fi
  od
}
```

Solution
[6p,4p]
Implementation for (a)

```
active proctype filter () {
  byte x;
end:
  do
    :: C ? x ->
       if
         :: (x >= t1) -> O1 ! x
         :: (x >= t2 && x < t1 ) -> O2 ! x
         :: else -> skip;
       fi;
  od
}

active proctype listener() {
  byte y = 0;
end:
  do
    :: O1 ? y -> assert (y >= t1)
    :: O2 ? y -> assert (y >= t2 && y < t1)
  od
```

```
}
```

Full trace for (b)

```
Starting P with pid 0
Starting Q with pid 1
Starting Z with pid 2
  1: proc  1 (Q) line  17 "br.pml" (state -) [values: 1!msgB]
  1: proc  1 (Q) line  17 "br.pml" (state 1) [C2!msgB]

  2: proc  2 (Z) line  29 "br.pml" (state -) [values: 1?msgB]
  2: proc  2 (Z) line  29 "br.pml" (state 6) [C2?x]
Z(2):x = msgB

  3: proc  2 (Z) line  30 "br.pml" (state -) [values: 2!msgB]
  3: proc  2 (Z) line  30 "br.pml" (state 7) [C1!x]
queue 2 (C1): [msgB]

  4: proc  2 (Z) line  25 "br.pml" (state -) [values: 2?msgB]
  4: proc  2 (Z) line  25 "br.pml" (state 1) [C1?x]
queue 2 (C1):
Z(2):x = msgB

  5: proc  0 (P) line  12 "br.pml" (state -) [values: 2!msgA]
  5: proc  0 (P) line  12 "br.pml" (state 1) [C1!msgA]
queue 2 (C1): [msgA]

spin: trail ends after 5 steps
#processes: 3
queue 2 (C1): [msgA]
  5: proc  2 (Z) line  26 "br.pml" (state 4)
  5: proc  1 (Q) line  18 "br.pml" (state 2)
  5: proc  0 (P) line  14 "br.pml" (state 2)
3 processes created
Exit-Status 0
null
```

Assignment 2 (Temporal Logic) (10p)

Consider the following PROMELA model:

```
byte x = 0;
bool b = false

active proctype P() {
  do
    :: x < 20 -> x = 20; b = true
    :: x >= 0 -> if
                   :: x < 30 -> x++
                   :: else -> x = 10
                 fi
  od
}
```

Take your time to understand the behavior of `P`. Then consider the following properties, each of which *might or might not* hold:

1. `b` will be **true** at some point.
2. `x` will always be ≥ 10.
3. At some point, `x` will be 10.
4. At some point, `x` will be 11.
5. From some point on, `x` will always be ≥ 10.
6. `x` will infinitely often be 11.
7. If `b` will never be **true**, then `x` will infinitely often be 11.

(a) Formulate each of the properties 1. - 7. in Temporal Logic.

(b) For each of the properties 1. - 7., tell whether or not the property is valid in the transition system given by the above PROMELA model. (You don't need to explain your answer.)

Solution
[6p, 4p]
(a)

1. `<>b`
2. `[](x >= 10)`
3. `<>(x == 10)`
4. `<>(x == 11)`

5. `<>[](x >= 10)`
6. `[]<>(x == 11)`
7. `(!<>b) -> []<>(x == 11)`

(b)

1. *invalid*
2. *invalid*
3. *valid*
4. *invalid*
5. *valid*
6. *invalid*
7. *valid*