Ex5

s1260119 Yuta Tomiyama

Task 5.1

```
int stick1 = 1
int stick2 = 1

active [2] proctype P1() {
  atomic {
    stick1 > 0
    stick1--
  }
  atomic {
    stick2 > 0
    stick2--
  }
  stick1++;
  stick2++;
}
```

Task 5.2

1

```
int Semaphore = 2;            // array sorting
bool sorted1 = false;   // (previous lecture)
bool sorted2 = false;
bool merging = false;

active proctype SortFirstHalf()
    { sorted1 = true; Semaphore--; }

active proctype SortSecondHalf()
    { sorted2 = true; Semaphore--; }

active proctype Merging() {
  Semaphore == 0
  merging = true
  assert(sorted1)
  assert(sorted2)
}
```

2

```
int stick1 = 1
int stick2 = 1
int Semaphore = 2
active [2] proctype P1() {
  atomic {
    stick1 > 0
    stick1--
  }
  atomic {
    stick2 > 0
    stick2--
  }
  stick1++;
  stick2++;
  Semaphore--
}

active proctype main() {
  Semaphore == 0
  assert(stick1 == 1)
  assert(stick2 == 1)
}
```

Task 5.3

```
ltl {
  (Semaphore == 2 -> <>(Semaphore == 1))
  (Semaphore == 1 -> <>(Semaphore == 0))
}

int Semaphore = 2; int Account = 0;

active [2] proctype Spouse(){
    int i = 0;
    do
    :: i >= 10 -> break;
    :: else -> Account++; i++;
    od;
    Semaphore--;
}

active proctype main(){
    Semaphore == 0;
    printf("Account = %d\n", Account);
    assert(Account == 20)
}
```

Task 5.4

```
bool doors_open = true;
int state = 1;
```

```
// 1 = down, 2 = moving up, 3 = up, 4 = moving down

ltl {
  (state == 2 || state == 4) U !doors_open
  (state == 3 -> <>(doors_open))
  (state == 3 -> <>(state == 1 && doors_open))
}

active proctype main()
{
do
:: state == 1 && doors_open -> doors_open = false; state = 2;
:: state == 2 -> state = 3;
:: state == 3 && !doors_open -> doors_open = true;
:: state == 3 && doors_open -> doors_open = false; state = 4;
:: state == 4 -> state = 1;
:: state == 1 && !doors_open -> doors_open = true;
od;
}
```