T-106.420 Concurrent programming
Example solution for homework assignment 3

1. (a) If the *Bear* process executes *P(mutex)* before a *Bee* process, we end up in a deadlock, since mutex is "down", and the bear is "sleeping".

   (b) Both semaphores are initialized to 0, therefore regardless of which process starts executing first, they both end up in a deadlock.

   (c) A working solution *(with passing the baton)*:

```
sem full = 0;
sem pot  = 1;
int portions = 0;

process Bear {
  while (true) {
    P(full);
    portions = 0;
    V(pot);
  }
}

process Bee[i = 1 to N] {
  while (true) {
    P(pot);
    portions++;

    if (portions == H)
      V(full);
    else
      V(pot);
  }
}
```

2. This solution assumes that DEC returns the new value:

```
/* P(s): */
while (s <= 0);
while (DEC(s) < 0) {
  INC(s); # undo decrement
  while (s<=0);  # wait until DEC could succeed
}

/* V(s): */
INC(s);
```

3. Monitor implementation of a semaphore:

```
monitor Sem {       /* begin monitor Sem */
  int   s = 0;
  cond  c;

  procedure P() {   /* P operation */
    while (s == 0)
      wait(c);

    s = s - 1;
  }

  procedure V() {   /* V operation */
    s = s + 1;
    signal(c);
  }

}   /* end monitor Sem */
```