Quiz

- What if we reverse the order of the first two lines the 2-process Peterson's algorithm

```
P0:                          P1:
turn = 1;                    turn = 0;
flag[0] = true;              flag[1] = true;
                             …
```

  Would it work?

- Prove that Peterson's N-process algorithm ensures:
  - ✓mutual exclusion: no two processes are in the critical section at a time
  - ✓starvation freedom: every process in the trying section eventually reaches the critical section (assuming no process fails in the trying, critical, or exit sections)
- Show that the bounded (black-white) Bakery algorithm in correct

Bakery [Lamport'74,original]

```
// initialization
flag: array [1..N] of bool = {false};
label: array [1..N] of integer = {0}; //assume no bound

// code for process i that wishes to enter CS
flag[i] = true; //enter the doorway
label[i] = 1 + max(label[1], ..., lebel[N]); //pick a ticket
flag[i] = false; //exit the doorway
for j=1 to N do
          while (flag[j]); //wait until j is not in the doorway
          while (label[j]≠0 and (label[j],j)<<(label[i],i));
          // wait until j is not "ahead"
…
// critical section
…
label[i] = 0; // exit section
```

Ticket withdrawal is "protected" with flags: a very useful trick

Black-White Bakery [Taubenfeld'06]

```
// initialization
color: {black,white};
flag: array [1..N] of bool = {false};
label[1..N]: array of type {0,…,n} = {0}  //bounded ticket numbers
mycolor[1..N]: array of type {black,white}

// code for process i that wishes to enter CS
flag[i] = true; //enter the "doorway"
mycolor[i] =color;
label[i] = 1 + max({label[j]| j=1,…,N: mycolor[i]=mycolor[j]});
flag[i] = false; //exit the "doorway"
for j=1 to N do
   while (flag[j]);
   if mycolor[j]=mycolor[i] then
      while (label[j]≠0 and (label[j],j)<<(label[i],i) and mycolor[j]=mycolor[i]  );
   else
      while (label[j]≠0 and mycolor[i]=color and mycolor[j] ≠ mycolor[i]);

// wait until all processes "ahead" of my color are served
…
// critical section
…
If mycolor[i]=black then color = white else color – black;
label[i] = 0; // exit section
```

Colored tickets => bounded variables!