In-class Exercise: The code below shows a function `resume()` similar to the one found in Xinu. The function captures a process' priority in local variable prio before making the process eligible to use the CPU (i.e., before calling `ready()`). The calls to `disable()` and `restore()` disable CPU interrupts and restore them. If the two statements with comments `STATEMENT A` and `STATEMENT B` are reversed, `resume` can return a priority value that the resumed process never had. Briefly explain how. *(Acknowledgment: Prof. Comer designed this problem.)*

```
/*-----------------------------------------------------------------------
* resume - make a process ready and return its priority
*---------------------------------------------------------------------*/
pri16 resume(
      pid32 pid                     /* ID of process to make ready */
      )
{
      intmask mask;                 /* saved interrupt mask */
      pri16 prio;                   /* priority value to return */
      mask = disable();             /* disable interrupts */

      /* Code is omitted that checks the argument; assume pid valid */

      /* Extract process' priority from the process table */
      prio = proctab[pid].pprio;    /* STATEMENT A */

      /* Make process ready and allow it to run */
      ready(pid, RESCHED_YES);      /* STATEMENT B */
      restore(mask);
      return prio;
}
```