Open Systems Seminar: Assignment 3
DUE: Wednesday May 23th

For this assignment, you'll start writing your own checkers. It has three parts:

1. Write a checker that catches when Linux code disables interrupts and calls either (1) `dev_kfree_skb` (it should call (`dev_kfree_skb_irq`), (2) any copy-to/from-user routines (their name will contain `*_user`) since this is a security hole or (3) `kmalloc` with the `__GFP_WAIT` flag set. This flag is included in a variety of different `kmalloc` options:

   ```
                     /* include/linux/mm.h */
   #define GFP_BUFFER      (__GFP_HIGH | __GFP_WAIT)
   #define GFP_ATOMIC      (__GFP_HIGH)
   #define GFP_USER        (             __GFP_WAIT | __GFP_IO)
   #define GFP_HIGHUSER    (             __GFP_WAIT | __GFP_IO | __GFP_HIGHMEM)
   #define GFP_KERNEL      (__GFP_HIGH | __GFP_WAIT | __GFP_IO)
   #define GFP_NFS         (__GFP_HIGH | __GFP_WAIT | __GFP_IO)
   #define GFP_KSWAPD      (                          __GFP_IO)
   ```

   To find this you will need to use MC routines to extract the integer flag value from out of the `kmalloc` call.

2. Fix two errors of different types and generate patches. For `kmalloc` this should be a simple matter of switching the flag to `GFP_ATOMIC`.

3. Come up with and implement one other rule or a deriver that will derive instances of a general rule (e.g., cases of "A must be paired with B").

   Possible rules:

   - Check that a set of functions are rolled back on error paths. (You could derive such functions or use `grep`.)
   - Derive which variables and functions are protected by which locks or with interrupt enable/disable.
   - Check when inconsistent error codes are returned for events (e.g., `kmalloc` should usually produce `ENOMEM`).
   - Derive which non-pointer returning functions can fail.

- Write a deriver to find pairings of functions, where you just consider the first and last function in pairings of the form:

  ```
  foo(x, ...);
  bar(x, ...);  /* ignore this intervening call */
  baz(x, ...);
  ```

  In this example, your deriver should emit something like "foo is paired with baz." Remember to get counter examples (i.e., a single function that is not paired, or has a non-standard pairing).

  Write a script or program that will pull in your examples and counter examples and emit a table with these sorted.