Part C: Crash Safety
Journaling

For any operation which must write multiple disk blocks atomically...

1) Write new blocks into the log, rather than target place. Track what target is.
2) Once all blocks are in the log, mark the log as “committed”
3) Copy data from the log to where they should be
4) Clear the commit flag

On system boot, check the log. If not committed, do nothing. If so, redo the copy (copy is idempotent)
Step 1: “log_begin()”

Make sure the log is cleared
Step 2: “bwrite(data block 1)”

Write into the log, rather than the place in the inode/extents region we want it to go.

Also need to track the actual location of the data block so you know where to write logged blocks to on recovery!
Step 3: “bwrite(data block 2)”

Write into the log, rather than the place in the inode/extents region we want it to go.
Step 4: “log_commit()” [1]

Mark the log as “committed”
Step 5: “log_commit()” [2]

Copy the first block from log onto disk
Step 6: “log_commit()” [3]

Copy the second block from log onto disk
Done!

We have both data blocks 1 and 2 on disk - everything was successful.

For efficiency, we can zero out the commit flag so the system doesn’t try to redo this
Example: before commit—CRASH

On reboot...
There’s no commit in the log, so we should *not* copy anything to the disk

The Log

<table>
<thead>
<tr>
<th>Log Header</th>
<th>Data Block 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>commit = 0</td>
<td>...</td>
</tr>
</tbody>
</table>

The Disk
(Main Storage)
Example: after commit, before clear—CRASH

On reboot, we see that there is a commit flag.

We can then copy block 1 and 2 to disk—
even though DB1 was already copied over,
overwriting it with the same data is fine.
Where to Log?

It’s just blocks on disk, so you can put it anywhere you want (within reason)

After-bitmap, before-inodes is a pretty good place

You’ll need to update the superblock struct and mkfs.c (mkfs.c initializes the disk during the compiling process)
Log API

- The spec recommends designing an API for yourself for log operations:
  - `log_begin_tx()`: (optional) begin the process of a transaction
  - `log_write()`: wrapper function around normal block writes
  - `log_commit_tx()`: complete a transaction and write out the commit block
  - `log_apply()`: log playback when the system reboots and needs to check the log for disk consistency

  Where/when should this be called? (Hint: inspect `kernel/fs.c`)