Catching up

When people code together on Replit, everyone's code needs to be in sync. You have to see the same code as I do even though we're typing on different computers. The challenge is making sure we don't end up with a jumbled mess of text while we type together.

So in order to keep everyone's code in sync, Replit uses a method called **Operational Transformations, or OT**.

Think about OT like this: when you type, you can either *insert* text, *delete* text, or *move* your cursor to a new position (this is called *skip* in OT land). These actions are called **operations**, and they **transform** your document!

More concretely, you can look at an Operational Transformation as a function that takes in a document, a position within that document (like where your cursor is), and then either modifies the document at that position or skips to a new position.

Some examples:

1. - Input document: `""`
   - Starting cursor position: `0`
   - Operation: `{"op": "insert", "chars": "Hello, human!"}`
   - Output document: `"Hello, human!"`
   - Ending cursor position: `13`

2. - Input document: `"What is up?"`
   - Starting cursor position: `7`
   - Operation: `{"op": "delete", "count": 3}`
   - Output document: `"What is?"`
   - Ending cursor position: `7`

   Watch out: `delete` operations are applied forward while keeping the cursor in place. Crazy, we know.

3. - Input document: `"Nice!"`
   - Starting cursor position: `0`
   - Operation (1): `{"op": "skip", "count": 4}`
   - Operation (2): `{"op": "insert", "chars": " day"}`
   - Output document: `"Nice day!"`
   - Ending cursor position: `8`

   As you can see, this last example applies two transformations in a row.

What we want you to do

Back to keeping everyone in a multiplayer repl in sync. In the real world, we don't always work at the same time. Sometimes people leave then rejoin later, or lose their internet connection and come back online. When people rejoin, we need their client to **"catch up"** to the current state of the repl so everyone has the same code in front of them!

So when we catch up, we need to validate that a sequence of operational transformations will actually produce the most recent version of your code. It would be pretty terrible if I edited your file while you're gone, then you somehow end up with the wrong file contents when you rejoin later 😢

For this challenge, you're going to write the OT validation function. The function will take in a string for the stale file contents, a string containing the latest file contents, and a JSON string containing the operations. Your function should validate that the sequence of operations, when applied to the stale contents, produces the latest contents. If it does not, or if the sequence of operations is invalid, your function should return false.

Submitting

Please sign up for a Replit account if you have not already. **Create a new repl for this problem and do all your coding in that repl.**

We ask that you only spend around 30 minutes on this, and suggest setting a timer. When you're ready to share your repl with us, please click the Invite button, generate a join link, and then copy it and send it to us!

We value straightforward, working code over clever, not-quite-working code, so try to get something working right away, even if it doesn't cover all the possible cases. Good luck! We can't wait to hear from you.

Gotchas:

- You can't skip past the end of a string
- You can't delete past the end of a string
- Delete operations are applied forward while keeping the cursor in place

You may use any language! Please choose a language you are comfortable coding in quickly. Here are some examples in JavaScript:

```
function isValid(stale, latest, otjson) {
  // this is the part you will write!
}

isValid(
  'Repl.it uses operational transformations to keep everyone in a multiplayer
  'Repl.it uses operational transformations.',
  '[{"op": "skip", "count": 40}, {"op": "delete", "count": 47}]'
); // true

isValid(
  'Repl.it uses operational transformations to keep everyone in a multiplayer
  'Repl.it uses operational transformations.',
  '[{"op": "skip", "count": 45}, {"op": "delete", "count": 47}]'
); // false, delete past end
); // false, delete past end

isValid(
  'Repl.it uses operational transformations to keep everyone in a multiplayer
  'Repl.it uses operational transformations.',
  '[{"op": "skip", "count": 40}, {"op": "delete", "count": 47}, {"op": "skip"
); // false, skip past end

isValid(
  'Repl.it uses operational transformations to keep everyone in a multiplayer
  'We use operational transformations to keep everyone in a multiplayer repl
  '[{"op": "delete", "count": 7}, {"op": "insert", "chars": "We"}, {"op": "sk
); // true

isValid(
  'Repl.it uses operational transformations to keep everyone in a multiplayer
  'We can use operational transformations to keep everyone in a multiplayer r
  '[{"op": "delete", "count": 7}, {"op": "insert", "chars": "We"}, {"op": "sk
); // false

isValid(
  'Repl.it uses operational transformations to keep everyone in a multiplayer
  'Repl.it uses operational transformations to keep everyone in a multiplayer
  '[]'
); // true
```