for the third block etc.

    - `length` : the length of the block in bytes
        - This'll be `2^14` (16 * 1024) for all blocks except the last one.
        - The last block will contain `2^14` bytes or less, you'll need calculate this value using the piece length.
- Wait for a `piece` message for each block you've requested
    - The message id for `piece` is `7` .
    - The payload for this message consists of:
        - `index` : the zero-based piece index
        - `begin` : the zero-based byte offset within the piece
        - `block` : the data for the piece, usually `2^14` bytes long

After receiving blocks and combining them into pieces, you'll want to check the integrity of each piece by comparing it's hash with the piece hash value found in the torrent file.

Here's how the tester will execute your program:

```
$ ./your_bittorrent.sh download_piece -o /tmp/test-piece-0 sample.torrent 0
```

and here's the output it expects:

```
Piece 0 downloaded to /tmp/test-piece-0.
```

**Optional:** To improve download speeds, you can consider pipelining your requests. [BitTorrent Economics Paper] recommends having 5 requests pending at once, to avoid a delay between blocks being sent.