Validating Data

▶ *You can check your work here, or peek if you get stuck.*

With raw, line-by-line or string-based token-oriented input, a data loop only fails when it reaches end-of file. However, consider the `sumNumbers` filter, which you can open by clicking the little "running man" on the right.

This function reads and processes `doubles`, returning the sum. This function works fine when provided with a stream that contains nothing but `double`s. It **fails** when in cannot read a `double`; it also fails, of course, when it reaches end-of-file. Let's fix the function so that it processes **all** of the valid data in the file.

Stream Flags

All stream objects contain a set of *Boolean* variables, known as the **state flags**. You can check the value of these flags by calling one of the stream's member functions:

- `fail()` mean the stream is in the failed state. It will not accept any more input.
- `good()` means the stream is ready to read more input
- `eof()` means there is no more input for the stream to read.

When the stream object is placed in a **failed state**, **no error message is printed**; the rest of the input is simply not processed and the input stream stops working. To fix:

1. Read the stream while it is good. The easiest way is simply `while (in)`.
2. Only sum the number `if (in >> number)`
3. Otherwise, if you **haven't** reached `in.eof()`
   1. **Reset** the **error state**, by calling the member function: `cin.clear()`
   2. **Remove** the offending token from the stream with `in >> bad_data` where `bad_data` is a string.
   3. You may **print an error message** to `cerr`.

See if you can get it to work. You can check your work with the solution above, (or peek if you get stuck).