Lab 2: Corner Cases

1 Clarification

First, some clarification on things that are not made explicit in the handout.

1. For every line of input, your program must output something. Either a message indicating success (Table 2) or an error (Table 3). So for an empty line of input (nothing or just whitespace) you should print Error: invalid command.
2. Only the first error from the left should be reported per line of input. In the case of missing/extra arguments, these are errors in the arguments that are missing/extra and should be reported only if the preceding arguments were valid.

2 Extraction operator caveats

In this assignment we want to extract arguments from the input, where arguments are any non-whitespace characters separated by whitespace. The extraction operator (>>) almost works like this, except it doesn't always stop when it hits whitespace. Instead it stops when the next character can't be converted to the destination type. This behaviour leads to a few corner cases that you need to consider. It also leads to a number of tricky cases that you can ignore. These corner cases are outlined in the following section.

3 Corner cases

In general we will let the extraction operator define what a valid argument is. That is, if the extraction operator extracts the whole argument into the destination datatype and doesn't set the fail flag, then that argument was valid. We give some examples of valid and invalid arguments that may not be obvious below. We also specify which cases you can safely ignore. These cases will not be used for marking so your code does not need to deal with them.

3.1 Integer

When the destination is an integer, the extraction operator accepts from strings of digits, 0-9, optionally preceded by '-' or '+'. Floating point representations of integers are not accepted. It's not clear whether integer overflow should be an out of range error or invalid argument so you can **ignore** this case.

Argument	Valid?	Ignore?	Comment
`1.0`	No		Only '1' will be extracted
`1.`	No		Same as 1.0
`1e2`	No		Integer extraction does not accept scientific notation
`+`	No	Ignore	Hard to detect
`-`	No	Ignore	Hard to detect

Table 1: Examples of integer argument corner cases

3.2 Floating point

When the destination is a floating point datatype, the extraction operator may accept strings containing: digits, '+', '-', '.' and 'e'. The 'e' character can be used for specifying real numbers with scientific notation. This leads to many corner cases so for simplicity you can **ignore** the case where a floating point argument contains 'e'. However, your code should deal with floating point arguments containing other characters, in which case they should be invalid.

Argument	Valid?	Ignore?	Comment
`1.`	Yes		Same as 1.0
`.1`	Yes		Same as 0.1
`1e2`	Yes	Ignore	Ignoring use of 'e'
`1e`	No	Ignore	Hard to detect
`+`	No	Ignore	Hard to detect
`-`	No	Ignore	Hard to detect
`.`	No	Ignore	Hard to detect

Table 2: Examples of floating point argument corner cases

3.3 String

Any collection of non-whitespace characters is a valid string. The only exception is that a resistor name cannot be "all". The handout doesn't specify whether names are case sensitive so you can **ignore** cases where name is "ALL", "All", "aLL", etc.

3.4 End of File

When the end of file character is entered (ctrl-d) anywhere but the beginning of a new line it does not set cin.eof() so the template we provide will not exit. You can **ignore** this case and assume that the end of file character will always be at the beginning of a new line.