Title:	Methods and Tools for Software Engineering
Course ID:	ECE 650
WWW:	`https://ece.uwaterloo.ca/~agurfink/ece650/`
Slack	`https://uw-ece650.slack.com`
Lectures:	Friday, 08:30 – 11:20
Instructor:	Prof. Arie Gurfinkel, `first.last@uwaterloo.ca`, DC 2536
TA:	Jakub Kuderski, `first.last@uwaterloo.ca`

Office hours by appointment. Begin all email subjects with `[ECE650]`.

A simple calculator

In this example, we will implement a simple line calculator.

Our calculator works as follows:

- Take as inputs a series of calculator commands (add, subtract, multiply) and numbers;
- Use that input to compute the result of a calculation.

Sample input

The input comprises of lines each specifying a command. There are the following commands (1) *add* a number, (2) *subtract* a number, (3) *multiply* a number, (4) *divide* a number, (5) display the current result, (6) clear, i.e., set to zero, the current result.

```
=
0
+ 3
- 2
=
1
* 5
=
c
=
0
```

Commands

- Commands `+`, `-`, `*`, `/` are used to add, subtract, multiply, and divide, respectively. They are followed by an *integer* argument.
- `=` is used to display the current value computed by the calculator. The value is initially 0 and is changed by the commands above.
- `c` is used to reset the calculator by setting the current value to 0.

Input and Output

Your program should take input from standard input. Your program should output to the standard output. Error should be output to standard error. You can use exceptions in your code to catch errors.

Errors

The above example is that of a "perfect" user — someone that did not make any mistakes with specifying the input. You should account for errors in the input. If a line in the input is erroneous, you should immediately output an error message. The format of the message is to be the string "`Error:`" followed by a brief descriptive message about the error. For example:

```
+
Error: '+' command must be followed by an integer.
```

Your program should recover from the error as well. That is, your program should reject the errorneous line, but continue to accept input. Your program should not crash because of an error. Any erroneous input we try will be of a relatively benign nature that mimics honest mistakes a user makes. We will not try malicious input, such as unduly long lines or weird control characters.

A calculator with memory

Extend the calculator with an additional memory register by supporting the following additional commands:

- `m` copies the current result into the memory register
- `mc` clears the memory by setting the register to 0
- `mr` — *memory recall* — copies the content of the memory back into the current result
- the arithmetic commands (i.e., `+`, `-`, etc.) can take an optional parameter `m` to indicate that the value of the memory register should be used instead of a number.

An example interaction is given below:

```
=
0
+ 10
m
c
+ 4
=
4
+ m
=
14
```