Befeni Technical Test [Basic]

Test Overview

- The following problem is a simple test which offers you the opportunity to demonstrate some basic programming ability but also affords you a chance to show some of the other requirements that we ware looking for in our next developer.
- The basic problem should only take an hour or two to solve and the submitted code should be of the quality you would produce during a normal working week.
- Please provide instructions on how to run your code from the command line in the root of your submitted code in a `readme.md` file.
- You are welcome to use any external libraries you want. Include them in your submission.
- There is no requirement to cater for invalid input. But you are welcome to add this to your solution if you have time.

There are no tricks or gotchas: you only need to implement the functionality described below. It is a simple problem to solve, but code quality is as important as solving the problem itself.

As you will not have the opportunity to clarify requirements, feel free to note any assumptions in your submission.

Problem Description

Write some code to calculate a result from a set of instructions.

Instructions comprise of a keyword and a number that are separated by a space per line.

Instructions are loaded from file and results are output to the screen.

Any number of Instructions can be specified.

Instructions can be any binary operators of your choice (e.g., add, divide, subtract, multiply etc). The instructions will ignore mathematical precedence.

The last instruction should be `apply` and a number (e.g. `apply 3`). The calculator is then initialised with that number and the previous instructions are applied to that number.

Bonus Opportunities

As mentioned, this is a basic test that should not take long to complete! This should hopefully give you an opportunity to add a few more extra flourishes to your submission. Here is a brief idea of the sort of things you could include:

- Provide Unit Tests for your code
- Split the architecture into server / client and build a basic front end using your front end libraries of choice
- Dockerize your solution so that your code can be run with a simple `docker-compose up` command.

Examples of the calculator lifecycle

Example 1.

Input from file

```
add 2
multiply 3
apply 3
```

Output to screen

```
15
```

Explanation

```
(3 + 2) * 3 = 15
```

Example 2.

Input from file

```
multiply 9
apply 5
```

Output to screen

```
45
```

Explanation

```
5 * 9 = 45
```

Example 3.

Input from file

```
apply 1
```

Output to screen

```
1
```