# A Simple Calculator

Implement a calculator that can add, subtract, multiply and divide two numbers. The project can help you become familiar with the steps of C development, basic C syntax, and command-line commands. No GUI is needed for the calculator. A command-line program is fine.

## Special notes

- You are not expected to finish all the following requirements. Do not feel frustrated if you cannot finish them all. Just try your best and challenge your limit. Enjoy coding!
- Feel calm that you cannot find an explicit standard for a score of 100 because your manager/boss will also not give one when you are employed.

## Requirements

1. The programming language should only be C, **NOT** C++. Please save your source code into a `*.c` file, and compile it using a C compiler such as gcc (not g++). If you fail on this one, you cannot get a score over 60.
2. When you run the program as follows, it will print the expression and the result.

```
$./calculator 2 + 3
2 + 3 = 5
$./calculator 2 - 3
2 - 3 = -1
$./calculator 2 * 3
2 * 3 = 6
$./calculator 2 / 3
2 / 3 = 0.66666667
```

3. It can tell the reason why the operation cannot be carried out.

```
./calculator 3.14 / 0
A number cannot be divied by zero.
```

4. It can tell the reason when the input is not a number.

```
./calculator a * 2
The input cannot be interpret as numbers!
```

5. If you input some big numbers, what will happen? Please provide some possible solutions, and try to implement them.

```
./calculator 987654321 * 987654321 # The result should be
97546105778997104 1
```

```
./calculator 1.0e200 * 1.0e200 # The result should be 1.0e400
```

6. The expressions should be input through the command line arguments, like `$./calculator 2 + 3`. If no command line arguments are given, it will switch to another mode like

```
./calculator
2 + 3 # the input
2 + 3 = 5 # the output
2 * 3 # input another expression
2 * 3 = 6 # the output
quit # the command to quit
```

7. If you want to develop more features to make a better calculator, you can follow the calculator `bc`. Its manual can be found at https://www.gnu.org/software/bc/manual/html_mono/bc.html

## Rules

1. **Deadline**: Please submit your project report before its deadline. The deadline is **23:59, March 10**. After the deadline (even 1 second), **0 score!** No deadline extension for any students, even for those they enroll in the course late.

2. **Format**: Please submit the files as: *.pdf, xxx.c, Please do **NOT** put the files into a compressed one. To uncompress a lot of files will cost time for instructors. PDF format can make the layout of the report to be consistent on any computer.

3. **Report**: Your score will also depend on the quality of your source code **and your report**. Your report should be easy to understand and describe your work well, especially the highlights of your work. If you do not know how to write the report, just assume you are a developer in a company and are reporting to your manager that you have developed a good calculator.

4. **Code style**: Please pay more attention to your code style. This is not ACM-ICPC contest. You have enough time to write code with both correct results and a good code style. You will get a deduction if your code style is terrible. You can read Google C++ Style Guide (http://google.github.io/styleguide/cppguide.html) or some other guide for code style. You may lose points on your project if your code is ugly.

5. **ChatGPT**: You can use ChatGPT as your personal coach. Such as you can use ChatGPT or some other similar tools to polish the text in your report. But ChatGPT can also generate some rubbish that looks good but contains nothing. It will waste the time of the instructor to review those empty words, and also make you lose points on your project. ChatGPT can also generate a framework of the source code, and help improve the quality of your source code. If you use it for that, please mention it in the report. You are responsible for all materials you submit, not ChatGPT.