Practical 01

By the end of this set of exercises, you will be familiar with:

- `print` and `read`
- `if`
- `do`
- Basic types

All `programs` *must* contain `implicit none`. While not strictly necessary, it will eliminate an entire class of bugs.

It is a very good idea to always use the following compiler flags for all these problems: `-Wall -Wextra -fcheck=all -g`. Try to ensure you have no warnings. You may also want to use `-std=f2008` or `-std=f2018`, depending on the version of `gfortran` you are using. This will ensure you stick to standard Fortran.

You should use the lecture materials for help/inspiration, but please don't copy and paste! There is some value to be had in typing up the programs yourself.

There are very likely too many exercises here for the time you have available. It's ok if you don't get through them all!

There may be several ways to solve each problem. If you have time, you might like to try different approaches.

Use a separate file and program for each exercise.

Hello World!

1. Open a text file called `hello.f90`. Write a simple "hello world" program that prints a message to the screen. Include at least one comment. Use `gfortran hello.f90` to compile it to `a.out`. Run your program with `./a.out`. Check it does what you think it should.
2. Time to break the program! Delete the `p` in `program` and try to recompile. What happens? What does the error message say?
3. Undo the deletion. Delete another character (for example, the `h` in `hello`) and see if it breaks the program, and if so, how. How many unique error messages can you find from single-character deletions? Which characters don't matter?

Hello <name>!

1. Write up the `hello_input` program from the lectures into a new file, `hello_input.f90`. Compile it, this time using the `-o` flag to give the executable a name. Run the program and give it some input.
2. Try the following ways to break the program. For each method, try to explain why the program behaves the way it does.
    1. Enter two words when it asks your name
    2. Enter a single word longer than 20 characters
    3. Enter a number with a decimal point
3. Create a second `character(len=20)` variable, try reading the two `character` variables with a single `read`, and add your new variable to the `print`

Further

1. Fix the three problems in question 2 above

Summing integers

1. Write a program that sums all the integers from 1 to 100 using a loop. Hint: the correct answer is 5050
2. Modify the program so that it takes an integer from user input, and then sums all the integers up to that number using a loop. Hint: you can check your answer with the formula:

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

Further

1. What happens if the user supplies a negative number? Hint: try looping over the `read` until the number is acceptable.
2. Take two numbers from the user, and sum all the integers between those numbers.
3. Take three numbers from the user, and use the third number as stride. That is, take a, b, and n, and sum every n^{th} number between a and b.

Solving quadratics

The solutions to the quadratic $ax^2 + bx + c = 0$ are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

1. Take three real numbers from user input, and print the two solutions for x using the above formula. If there are no real solutions, print a message saying so.
2. Extend your program to also print complex solutions

Sample output:

```
$ gfortran quadratics.f90 -o quadratics
$ ./quadratics
 Enter a, b, c separated by spaces:
2 4 1
 Solutions are:
   x =  -1.70710683
   x = -0.292893231
```

Further

1. Can you extend this to solve cubics?
2. What about quartics?