- Befunge Instruction Sheet
  - `0` -> `9` - Add integer to Stack
  - `+` - Addition: Pop a and b, then push a+b
  - `-` - Subtraction: Pop a and b, then push b-a
  - `*` - Multiplication: Pop a and b, then push a*b
  - `/` - Integer division: Pop a and b, then push b/a, rounded towards 0.
  - `%` - Modulo: Pop a and b, then push the remainder of the integer division of b/a.
  - `:` - Duplicate value on top of the stack
  - \ - Swap two values on top of the stack
  - `$` - Pop value from the stack and discard it
  - `.` - Pop value and output as an integer followed by a space
  - `,` - Pop value and output as ASCII character
  - `@` - End of Program
  - `>` - Start moving RIGHT
  - `<` - Start moving LEFT
  - `^` - Start moving UP
  - `v` - Start moving DOWN
  - `#` - Bridge, jump over next instruction.
  - `!` - Logical NOT: Pop a value. If the value is zero, push 1; otherwise, push zero.
  - ` - Greater than: Pop a and b, then push 1 if b>a, otherwise zero.
  - `_` - Pop a value; move right if value=0, left otherwise
  - `|` - Pop a value; move down if value=0, up otherwise
  - `p` - Pop y, x, and v, then change the character at (x,y) in the program to the character with ASCII value v
  - `g` - Get a value from a grid element. Pop y and x, then push ASCII value of the character at that position in the program.

Build a Befunge Parser

- Use your preferred language to build a Befunge parser.

  - Given a program in a 2D Grid, your parser should navigate through the program & output the result.
  - Start by implementing a `simple` calculator which just reads across a single line & outputs a result.
  - For example, this calculation should give you a result of 42:

  ```
  32*81-*.@
  ```

Create Befunge Programs

- Extend your parser to handle loops and conditional statements.
    - Test your parser by creating a Befunge program which counts down from 9 until 1 before printing "Liftoff"
    - Create a Befunge program which calculates `x` to the power of `y` i.e. x=2, y=4 should give a result of 16. For simplicity x & y are both between 1 & 9.
    - Create a Befunge program which calculates the first 10 Fibonacci numbers.
    - Create a Befunge program which calculates `x` Factorial.

NB - If possible try & solve the above problems without using the `p` & `g` operators to update the Program Grid.