(1) You are a TA at a university, and you want to evaluate your student's homework without executing their (untrusted) code. You decide to write a small web-service that takes bytecode as input, and interprets the results.

The bytecode language you need to support includes basic arithmetic and variables. The bytecode language is stack, rather than register based. ByteCode (right) is given for the following pseudo code (left):

```
function f() {

    x = 1                LOAD_VAL 1
                       WRITE_VAR 'x'

    y = 2                LOAD_VAL 2
                       WRITE_VAR 'y'

    return (x + 1) * y     READ_VAR 'x'
                       LOAD_VAL 1
                       ADD

                       READ_VAR 'y'
                       MULTIPLY

                       RETURN_VALUE
}
```

Add a data type `ByteCode` that can represent bytecode like in the example above, along with an interpreter for said bytecode. Make sure your bytecode is flat, i.e. not nested.

(2) Extend your interpreter with loops. In particular:
   (a) Extend your `ByteCode` data type with suitable instructions to support loops
   (b) Modify your interpreter to support said instructions
   (c) Try it out and see if it works :)

(3) Suppose we added the following bytecode instructions to our language:
   SEND_CHANNEL:
      Pops the channel and a value from the stack and send the
      value on the channel using a blocking send

   RECV_CHANNEL:

Pops the channel from the stack, receives a value from the channel (this may block), and push the resulting value back onto the stack

SPAWN:
Pop two functions from the stack and spawn them as concurrent tasks

Describe in a few sentences how each bytecode instruction could be interpreted, and how your interpreter or language runtime could deal with the blocking nature of the send and the receive instructions.

(4)
Write a function that given a directory, recursively finds all files with a given file extension in that directory and all sub-directories, and counts the number of lines in the file and prints it to stdout.

Blockchain Questions (optional, only if you have previous blockchain experience)

(5) explain some of the ways hashing functions enable blockchain technology

(6) briefly explain Bitcoin's UTXO model of transaction validation (separate from POW)

(7) what is the structure of a Block in bitcoin and how does it relate to the 'blockchain' (merkle tree vs merkle list of merkle trees)

(8) what problem/s are POW/POS trying to solve? discuss/compare (byzantine fault tolerance, reaching a single consensus on a p2p network)