EVM PUZZLES

Puzzle 1

```
00    34    CALLVALUE

01    56    JUMP

02    FD    REVERT

03    FD    REVERT

04    FD    REVERT

05    FD    REVERT

06    FD    REVERT

07    FD    REVERT

08    5B    JUMPDEST

09    00    STOP
```

Explanation: Input (CALLVALUE) is inputted into JUMP, we want to not revert. The only opcodes that do this are JUMPDEST and STOP. JUMPDEST is a valid JUMP destination, so we jump there, and the next opcode is STOP.

Solution: 8

Puzzle 2

```
00   34   CALLVALUE

01   38   CODESIZE

02   03   SUB

03   56   JUMP

04   FD   REVERT

05   FD   REVERT

06   5B   JUMPDEST

07   00   STOP

08   FD   REVERT

09   FD   REVERT
```

Explanation:

We want JUMP to have 6 as an input. JUMP's input will be SUB's output. SUB will subtract the second value on the stack from the first. The first value is our input, the second is CODESIZE which is equal to 10 (the current running contract's codesize). Maybe 10 since there are 10 opcodes in the contract. So were solving 10 – solution = 6, solution = 4!

Solution: 4

Puzzle 3

```
00    36    CALLDATASIZE
01    56    JUMP
02    FD    REVERT
03    FD    REVERT
04    5B    JUMPDEST
05    00    STOP
```

Explanation:

Want CALLDATASIZE = 4, so lets input 00000000 (4 bytes). Success!

Solution: 00000000

Puzzle 4

```
00    34    CALLVALUE

01    38    CODESIZE

02    18    XOR

03    56    JUMP

04    FD    REVERT

05    FD    REVERT

06    FD    REVERT

07    FD    REVERT

08    FD    REVERT

09    FD    REVERT

0A    5B    JUMPDEST

0B    00    STOP
```

Explanation: Once again, we just need JUMP to jump to JUMPDEST, input needs to be 0A (10). We know CODESIZE = 12, so 12 XOR solution = 10. 12 in binary is 00001100, 10 is 00001010. Solution = 00001100 XOR 00001010 = 00000110 = 6!

Solution: 6

Puzzle 5

```
00    34        CALLVALUE
01    80        DUP1
02    02        MUL
03    610100    PUSH2 0100
06    14        EQ
07    600C      PUSH1 0C
09    57        JUMPI
0A    FD        REVERT
0B    FD        REVERT
0C    5B        JUMPDEST
0D    00        STOP
0E    FD        REVERT
0F    FD        REVERT
```

Explanation: Same deal, except its JUMPI, inputs are destination, bool. Assuming >0 bool is considered true. Dest will be result of PUSH1 0C which is exactly where we want to go but we have a EQ output as the bool, so we need MUL output to equal 0100. MUL takes in output of DUP1 which is just CALLVALUE, and CALLVALUE. So solve solution^2 = 0100. 0100 = 4. Square root of 4, easy, 2! Wrong. :(. Bytecode speaks in hex, not binary. 0100 = 1 + ff = 1 + 255 = 256. Square root of 256, less easy, but 16!

Solution: 16

Puzzle 6

```
00     6000     PUSH1 00
02     35       CALLDATALOAD
03     56       JUMP
04     FD       REVERT
05     FD       REVERT
06     FD       REVERT
07     FD       REVERT
08     FD       REVERT
09     FD       REVERT
0A     5B       JUMPDEST
0B     00       STOP
```

Explanation: ooo, what does CALLDATALOAD do? Returns data[input]. So we need to input data array where the first element is 10. Hmm 0A? No. How does data look in hex? Maybe needs to be 4 bytes? 0A000000? Nope. Ok im busting out evm.codes. Inputting 0A as calldata gives me "[Error] Calldata should be a hexadecimal string with 2 digits per byte". Ok… "0a"? no. Ok I googled the error. Needs to be 40 digits? 0x0a? something something big endian? 0xaaaaaacccccccccccccccccccccccccccccccccccccaaaaaaaaaaaaaaaaaaaaaaaa0a? noooo. Idgi

0x000000000000000000000000000000000000000000000000000000000a

[Error] Invalid JUMP at c970b6212d9176bae15f0af167edc3aa095e72ec64471cee9f67b3769d0ab03c

I need to input 0xa

No 0x000000000000000000000000000000000000000000000000000000000000000a

Bruh

Easy

Just needs to be 10 in uint256, so pad zeros, get those fingers movin'!

Solution: 0x000000000000000000000000000000000000000000000000000000000000000a

Puzzle 7

```
00    36      CALLDATASIZE //20?
01    6000      PUSH1 00 //00 20
03    80      DUP1 // 00 00 20
04    37      CALLDATACOPY // memory
05    36      CALLDATASIZE
06    6000      PUSH1 00
08    6000      PUSH1 00
0A    F0      CREATE // 00 00 20
0B    3B      EXTCODESIZE
0C    6001      PUSH1 01
0E    14      EQ
0F    6013      PUSH1 13
11    57      JUMPI
12    FD      REVERT
13    5B      JUMPDEST
14    00      STOP
```

Explanation: Ok im going to have to figure this one out then type lol. Brb.
Need to input calldata that has 01 codesize when deployed as a contract…

Same solution as puzzle 6? No. calldata needs to publish a contract, something about sending creation code to zero address.

Googled "what is the smallest bytecode that will publish a cont ract" ezpz 0x600180600b6000396000f3, can also use the code values of this contract, nvm not a full contract.

Requirements for deploying contract: Call RETURN with stack [0 non-zero] and memory [something]

Solution: 0x600180600b6000396000f3

Puzzle 8 Legend: [stack] {memory}

```
00   36     CALLDATASIZE     [N] {}
01   6000   PUSH1 00         [0 N] {}
03   80     DUP1             [0 0 N] {}
04   37     CALLDATACOPY     [] { CALLDATA }
05   36     CALLDATASIZE     [N] { CALLDATA }
06   6000   PUSH1 00         [0 N] { CALLDATA }
08   6000   PUSH1 00         [0 0 N] { CALLDATA }
0A   F0     CREATE           [ADDRESS] { CALLDATA }
0B   6000   PUSH1 00         [0 ADDRESS] { CALLDATA }
0D   80     DUP1             [0 0 ADDRESS] { CALLDATA }
0E   80     DUP1             [0 0 0 ADDRESS] { CALLDATA }
0F   80     DUP1             [0 0 0 0 ADDRESS] { CALLDATA }
10   80     DUP1             [0 0 0 0 0 ADDRESS] { CALLDATA }
11   94     SWAP5            [ADDRESS 0 0 0 0 0] { CALLDATA }
12   5A     GAS              [GAS ADDRESS 0 0 0 0 0] { CALLDATA }
13   F1     CALL // needs to return 0, takes 7 inputs. Call returns 0 when… reverts, 1 if
ADDRESS is not contract, so CALLDATA just needs to deploy a contract
14   6000   PUSH1 00
16   14     EQ // needs to be true
17   601B   PUSH1 1B
19   57     JUMPI
1A   FD     REVERT
1B   5B     JUMPDEST
1C   00     STOP
```

Explanation: Read comments next to opcodes, used 3859818153F3 from the stackexchange Ethereum post here https://ethereum.stackexchange.com/questions/40757/what-is-the-shortest-bytecode-that-will-publish-a-contract-with-non-zero-bytecod

Puzzle 9

```
00    36      CALLDATASIZE
01    6003      PUSH1 03
03    10      LT // 03 < CALLDATASIZE
04    6009      PUSH1 09
06    57      JUMPI
07    FD      REVERT
08    FD      REVERT
09    5B      JUMPDEST
0A    34      CALLVALUE
0B    36      CALLDATASIZE
0C    02      MUL // CALLDATASIZE * CALLVALUE = 8, CALLDATASIZE > 3, so CALLDATASIZE = 4
0D    6008      PUSH1 08
0F    14      EQ
10    6014      PUSH1 14
12    57      JUMPI
13    FD      REVERT
14    5B      JUMPDEST
15    00      STOP
```

Explanation: starting from the end, need output of MUL to equal 8, inputs are CALLDATASIZE and CALLVALUE, before that need to get to JUMPDEST, so LT must be true and that takes 03 and CALLDATASIZE as inputs, and checks that first is less than second so 3 < CALLDATASIZE. Solving this we get CALLDATASIZE = 4 and CALLVALUE = 2 ☺ used 0xffffffff as CALLDATA

Solution: CALLVALUE: CALLDATASIZE: 0xffffffff

Puzzle 10

00 38 CODESIZE

01 34 CALLVALUE

02 90 SWAP1

03 11 GT // needs to be true, CODESIZE > CALLVALUE = 15, CODESIZE = 1A = 26 so its true, so just callvalue 15?

04 6008 PUSH1 08

06 57 JUMPI

07 FD REVERT

08 5B JUMPDEST

09 36 CALLDATASIZE [CALLDATASIZE]

0A 610003 PUSH2 0003 [0003 CALLDATASIZE]

0D 90 SWAP1 [CALLDATASIZE 0003]

0E 06 MOD // first divided by second, CALLDATASIZE = 3, CALLDATA = 0xffffff

0F 15 ISZERO // needs to be true for JUMPI on 14,

10 34 CALLVALUE

11 600A PUSH1 0A 10 + WHAT =

13 01 ADD

14 57 JUMPI // need to jump to 19, 0a + WHAT = 19(hex) = 25 ? Well… 9 + 16 –10= 15,CALLVALUE=15

15 FD REVERT

16 FD REVERT

17 FD REVERT

18 FD REVERT

19 5B JUMPDEST

```
1A     00        STOP
```