

Hurdles (Part 1)

CSCG is in may ways like hurdling. Our challenge attempts to hide the obstacles in a way you might not be used to. Can you solve the first two stages? Difficulty: Medium

Hurdles was a two part reversing challenge from the CSCG Qualifications 2023. This writeup is for the first part.

Overview (Ghidra)

Simply running the program gives:

```
$ ./hurdles
Bad input
```

Opening the binary in Ghidra we can get an overview of the flag checker. The main function decompiles as follows (function names were changed):

```
undefined8 main(int argc, char **argv) {
    char cVar1;
    uint uVar2;
    undefined8 uVar3;
    ulong uVar4;

    cVar1 = check_arg_and_length();
    if ((cVar1 != '\0') && (uVar2 = stage1(argc, argv), (char)uVar2 != '\0')) {
        puts("You have completed stage 1");
        uVar3 = stage2(argc, argv);
        if ((char)uVar3 != '\0') {
            puts("You have completed stage 2");
            FUN_0048ab10(argc, argv);
            uVar4 = stage3(argc, argv);
            if ((char)uVar4 != '\0') {
                puts("You have completed stage 3");
                cVar1 = stage4(argc, argv);
                if (cVar1 != '\0') {
                    puts("You have completed stage 4");
                    printf("Here is your flag: cscg{%s}\n", argv[1]);
                    return 0;
                }
            }
        }
    }
    puts("Bad input");
    return 0xffffffff;
}
```

check_arg_and_length does only very basic checks:

```
ulong check_arg_and_length(int argc, char **argv) {
    ulong uVar1;

    if (argc == 2) {
        uVar1 = strlen(argv[1]);
        return uVar1 & 0xffffffffffff00 | (ulong)(uVar1 < 0x23);
    }
    return 0;
}
```

strlen is easily identifiable, because of it's decompilation.

That means, in order to pass check_arg_and_length we have to supply one argument to the program, which has to be shorter than 0x23 characters.

Stage 1 (Angr)

Next up is stage1, which can actually still be reversed using Ghidra, but let's use [angr](#), as it will be helpful during the rest of the challenge. More precisely, I will be using angrs symbolic execution feature. You can imagine it like a simulator, which discovers restrictions on your input, that have to be fulfilled, to reach certain code paths. It can then also generate possible inputs which match these requirements (much like z3).

To solve stage1 we can use the following script:

```
import angr
import claripy

proj = angr.Project("./hurdles", auto_load_libs=False)

proj.hook(0x06ef028, angr.SIM_PROCEDURES['glibc']['__libc_start_main']())
proj.hook(0x06ef038, angr.SIM_PROCEDURES['libc']['free']())
proj.hook(0x06ef020, angr.SIM_PROCEDURES['libc']['malloc']())
proj.hook(0x06ef000, angr.SIM_PROCEDURES['libc']['printf']())
proj.hook(0x06ef010, angr.SIM_PROCEDURES['libc']['puts']())

arg1 = claripy.BVS('arg1', 8*0x22)

initial_state = proj.factory.entry_state(args=["./hurdles", arg1])

sm = proj.factory.simulation_manager(initial_state)
sm.explore(find=0x0400834, avoid=0x040089f)

found = sm.found[0]

print(found.solver.eval(arg1, cast_to=bytes))
```

It first loads the `hurdles` executable, but without loading external libs like `libc` . That means we have to hook the `libc` functions with python code that emulates the behavior. This is done to avoid tracking unnecessary restrictions on the input and to keep the amount of simulated instructions low.

Then we define our input as an unrestricted vector of `0x22` bytes, which can be used to build an simulation state, that executes the binary from the entry point.

Then we simulate the code until we hit the address `0x0400834` (`puts("You have completed stage 1")`) without hitting `0x040089f` (`puts("Bad input");`).

We will find a simulation state, that matches that criterion. Evaluating the conditions on `arg1` leading to this code path can be done using the states `solver` attribute.

In the end the script prints: `b'1_kn0w_h0w_\x00'`

Running `./hurdles 1_kn0w_h0w_` , we can check that we have actually passed stage 1.

Stage 2 (Angr + GDB)

Sadly we cannot simply copy the strategy from `stage1` to `stage2` .

```
import angr
import claripy

proj = angr.Project("./hurdles", auto_load_libs=False)

proj.hook(0x06ef028, angr.SIM_PROCEDURES['glibc']['__libc_start_main']())
proj.hook(0x06ef038, angr.SIM_PROCEDURES['libc']['free']())
proj.hook(0x06ef020, angr.SIM_PROCEDURES['libc']['malloc']())
proj.hook(0x06ef000, angr.SIM_PROCEDURES['libc']['printf']())
proj.hook(0x06ef010, angr.SIM_PROCEDURES['libc']['puts']())

# Further optimization: make strlen simply return 0x22 and bypass `check_arg_and_length`
proj.hook(0x04007e0, angr.SIM_PROCEDURES['stubs']['ReturnUnconstrained'](return_val=0x22))
proj.hook(0x04008c0, angr.SIM_PROCEDURES['stubs']['ReturnUnconstrained'](return_val=0x1))

arg1 = claripy.BVS('arg1', 8*0x22)

initial_state = proj.factory.entry_state(args=["./hurdles", arg1])

sm = proj.factory.simulation_manager(initial_state)
sm.explore(find=0x0400834, avoid=0x040089f).unstash(from_stash='found', to_stash='active')
sm.explore(find=0x040084c, avoid=0x040089f)

print(sm)
```

Executing this script yields `<SimulationManager with 16 avoid>` , so no code paths, that reach `0x040084c` (`puts("You have completed stage 2")`), have been found.

Decompiling the `stage2` function in Ghidra shows, that the following check has to pass in order for the function to return true.

```
// [...]
uVar2 = CONCAT71(0xa6e3a29dbfb830,
                 *(short *)(((ulong)(byte)flag[0xb] * 1000 + 0x2c8e2eb120231781 +
                             (ulong)(byte)flag[0xc] * 100 + (ulong)(byte)flag[0xd] * 10
                             + (ulong)bVar3) * 2 + -0x591c5d623fff17e2) == 0x3419);

// [...]
return uVar2;
```

Assembly:

```
00401d31  CMP             word ptr [RAX + R8*0x2 + 0x048b7c0],CX
00401d3a  SETZ           AL
// [...]
00402103  RET
```

So let's try to find a code path to the address of the instruction, that is responsible for this compare (`0x0401d31`) with angr. Indeed a path is found and if we evaluate the restrictions on the input, we get:

```
>>> sm
<SimulationManager with 1 found, 15 avoid>
>>> found = sm.found[0]
>>> found.solver.eval(arg1, cast_to=bytes)
b'1_kn0w_h0w_8012\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

Generating further examples is also possible:

```
>>> found.solver.eval_upto(arg1[-15 * 8:], 10, cast_to=bytes)
[b'1_kn0w_h0w_6090', b'1_kn0w_h0w_8810', b'1_kn0w_h0w_8490', b'1_kn0w_h0w_6010', b'1_kn0w_h0w_8850', b'1_kn0w_h0w_0011', b'1_kn0w_h0w_8090', b'1_kn0w_h0w_8830', b'1_kn0w_h0w_8870', b'1_kn0w_h0w_8890', b'1_kn0w_h0w_8910', b'1_kn0w_h0w_8930', b'1_kn0w_h0w_8950', b'1_kn0w_h0w_8970', b'1_kn0w_h0w_8990', b'1_kn0w_h0w_9010', b'1_kn0w_h0w_9030', b'1_kn0w_h0w_9050', b'1_kn0w_h0w_9070', b'1_kn0w_h0w_9090', b'1_kn0w_h0w_9110', b'1_kn0w_h0w_9130', b'1_kn0w_h0w_9150', b'1_kn0w_h0w_9170', b'1_kn0w_h0w_9190', b'1_kn0w_h0w_9210', b'1_kn0w_h0w_9230', b'1_kn0w_h0w_9250', b'1_kn0w_h0w_9270', b'1_kn0w_h0w_9290', b'1_kn0w_h0w_9310', b'1_kn0w_h0w_9330', b'1_kn0w_h0w_9350', b'1_kn0w_h0w_9370', b'1_kn0w_h0w_9390', b'1_kn0w_h0w_9410', b'1_kn0w_h0w_9430', b'1_kn0w_h0w_9450', b'1_kn0w_h0w_9470', b'1_kn0w_h0w_9490', b'1_kn0w_h0w_9510', b'1_kn0w_h0w_9530', b'1_kn0w_h0w_9550', b'1_kn0w_h0w_9570', b'1_kn0w_h0w_9590', b'1_kn0w_h0w_9610', b'1_kn0w_h0w_9630', b'1_kn0w_h0w_9650', b'1_kn0w_h0w_9670', b'1_kn0w_h0w_9690', b'1_kn0w_h0w_9710', b'1_kn0w_h0w_9730', b'1_kn0w_h0w_9750', b'1_kn0w_h0w_9770', b'1_kn0w_h0w_9790', b'1_kn0w_h0w_9810', b'1_kn0w_h0w_9830', b'1_kn0w_h0w_9850', b'1_kn0w_h0w_9870', b'1_kn0w_h0w_9890', b'1_kn0w_h0w_9910', b'1_kn0w_h0w_9930', b'1_kn0w_h0w_9950', b'1_kn0w_h0w_9970', b'1_kn0w_h0w_9990', b'1_kn0w_h0w_0000', b'1_kn0w_h0w_0001', b'1_kn0w_h0w_0002', b'1_kn0w_h0w_0003', b'1_kn0w_h0w_0004', b'1_kn0w_h0w_0005', b'1_kn0w_h0w_0006', b'1_kn0w_h0w_0007', b'1_kn0w_h0w_0008', b'1_kn0w_h0w_0009', b'1_kn0w_h0w_0010', b'1_kn0w_h0w_0011', b'1_kn0w_h0w_0012', b'1_kn0w_h0w_0013', b'1_kn0w_h0w_0014', b'1_kn0w_h0w_0015', b'1_kn0w_h0w_0016', b'1_kn0w_h0w_0017', b'1_kn0w_h0w_0018', b'1_kn0w_h0w_0019', b'1_kn0w_h0w_0020', b'1_kn0w_h0w_0021', b'1_kn0w_h0w_0022', b'1_kn0w_h0w_0023', b'1_kn0w_h0w_0024', b'1_kn0w_h0w_0025', b'1_kn0w_h0w_0026', b'1_kn0w_h0w_0027', b'1_kn0w_h0w_0028', b'1_kn0w_h0w_0029', b'1_kn0w_h0w_0030', b'1_kn0w_h0w_0031', b'1_kn0w_h0w_0032', b'1_kn0w_h0w_0033', b'1_kn0w_h0w_0034', b'1_kn0w_h0w_0035', b'1_kn0w_h0w_0036', b'1_kn0w_h0w_0037', b'1_kn0w_h0w_0038', b'1_kn0w_h0w_0039', b'1_kn0w_h0w_0040', b'1_kn0w_h0w_0041', b'1_kn0w_h0w_0042', b'1_kn0w_h0w_0043', b'1_kn0w_h0w_0044', b'1_kn0w_h0w_0045', b'1_kn0w_h0w_0046', b'1_kn0w_h0w_0047', b'1_kn0w_h0w_0048', b'1_kn0w_h0w_0049', b'1_kn0w_h0w_0050', b'1_kn0w_h0w_0051', b'1_kn0w_h0w_0052', b'1_kn0w_h0w_0053', b'1_kn0w_h0w_0054', b'1_kn0w_h0w_0055', b'1_kn0w_h0w_0056', b'1_kn0w_h0w_0057', b'1_kn0w_h0w_0058', b'1_kn0w_h0w_0059', b'1_kn0w_h0w_0060', b'1_kn0w_h0w_0061', b'1_kn0w_h0w_0062', b'1_kn0w_h0w_0063', b'1_kn0w_h0w_0064', b'1_kn0w_h0w_0065', b'1_kn0w_h0w_0066', b'1_kn0w_h0w_0067', b'1_kn0w_h0w_0068', b'1_kn0w_h0w_0069', b'1_kn0w_h0w_0070', b'1_kn0w_h0w_0071', b'1_kn0w_h0w_0072', b'1_kn0w_h0w_0073', b'1_kn0w_h0w_0074', b'1_kn0w_h0w_0075', b'1_kn0w_h0w_0076', b'1_kn0w_h0w_0077', b'1_kn0w_h0w_0078', b'1_kn0w_h0w_0079', b'1_kn0w_h0w_0080', b'1_kn0w_h0w_0081', b'1_kn0w_h0w_0082', b'1_kn0w_h0w_0083', b'1_kn0w_h0w_0084', b'1_kn0w_h0w_0085', b'1_kn0w_h0w_0086', b'1_kn0w_h0w_0087', b'1_kn0w_h0w_0088', b'1_kn0w_h0w_0089', b'1_kn0w_h0w_0090', b'1_kn0w_h0w_0091', b'1_kn0w_h0w_0092', b'1_kn0w_h0w_0093', b'1_kn0w_h0w_0094', b'1_kn0w_h0w_0095', b'1_kn0w_h0w_0096', b'1_kn0w_h0w_0097', b'1_kn0w_h0w_0098', b'1_kn0w_h0w_0099', b'1_kn0w_h0w_0100', b'1_kn0w_h0w_0101', b'1_kn0w_h0w_0102', b'1_kn0w_h0w_0103', b'1_kn0w_h0w_0104', b'1_kn0w_h0w_0105', b'1_kn0w_h0w_0106', b'1_kn0w_h0w_0107', b'1_kn0w_h0w_0108', b'1_kn0w_h0w_0109', b'1_kn0w_h0w_0110', b'1_kn0w_h0w_0111', b'1_kn0w_h0w_0112', b'1_kn0w_h0w_0113', b'1_kn0w_h0w_0114', b'1_kn0w_h0w_0115', b'1_kn0w_h0w_0116', b'1_kn0w_h0w_0117', b'1_kn0w_h0w_0118', b'1_kn0w_h0w_0119', b'1_kn0w_h0w_0120', b'1_kn0w_h0w_0121', b'1_kn0w_h0w_0122', b'1_kn0w_h0w_0123', b'1_kn0w_h0w_0124', b'1_kn0w_h0w_0125', b'1_kn0w_h0w_0126', b'1_kn0w_h0w_0127', b'1_kn0w_h0w_0128', b'1_kn0w_h0w_0129', b'1_kn0w_h0w_0130', b'1_kn0w_h0w_0131', b'1_kn0w_h0w_0132', b'1_kn0w_h0w_0133', b'1_kn0w_h0w_0134', b'1_kn0w_h0w_0135', b'1_kn0w_h0w_0136', b'1_kn0w_h0w_0137', b'1_kn0w_h0w_0138', b'1_kn0w_h0w_0139', b'1_kn0w_h0w_0140', b'1_kn0w_h0w_0141', b'1_kn0w_h0w_0142', b'1_kn0w_h0w_0143', b'1_kn0w_h0w_0144', b'1_kn0w_h0w_0145', b'1_kn0w_h0w_0146', b'1_kn0w_h0w_0147', b'1_kn0w_h0w_0148', b'1_kn0w_h0w_0149', b'1_kn0w_h0w_0150', b'1_kn0w_h0w_0151', b'1_kn0w_h0w_0152', b'1_kn0w_h0w_0153', b'1_kn0w_h0w_0154', b'1_kn0w_h0w_0155', b'1_kn0w_h0w_0156', b'1_kn0w_h0w_0157', b'1_kn0w_h0w_0158', b'1_kn0w_h0w_0159', b'1_kn0w_h0w_0160', b'1_kn0w_h0w_0161', b'1_kn0w_h0w_0162', b'1_kn0w_h0w_0163', b'1_kn0w_h0w_0164', b'1_kn0w_h0w_0165', b'1_kn0w_h0w_0166', b'1_kn0w_h0w_0167', b'1_kn0w_h0w_0168', b'1_kn0w_h0w_0169', b'1_kn0w_h0w_0170', b'1_kn0w_h0w_0171', b'1_kn0w_h0w_0172', b'1_kn0w_h0w_0173', b'1_kn0w_h0w_0174', b'1_kn0w_h0w_0175', b'1_kn0w_h0w_0176', b'1_kn0w_h0w_0177', b'1_kn0w_h0w_0178', b'1_kn0w_h0w_0179', b'1_kn0w_h0w_0180', b'1_kn0w_h0w_0181', b'1_kn0w_h0w_0182', b'1_kn0w_h0w_0183', b'1_kn0w_h0w_0184', b'1_kn0w_h0w_0185', b'1_kn0w_h0w_0186', b'1_kn0w_h0w_0187', b'1_kn0w_h0w_0188', b'1_kn0w_h0w_0189', b'1_kn0w_h0w_0190', b'1_kn0w_h0w_0191', b'1_kn0w_h0w_0192', b'1_kn0w_h0w_0193', b'1_kn0w_h0w_0194', b'1_kn0w_h0w_0195', b'1_kn0w_h0w_0196', b'1_kn0w_h0w_0197', b'1_kn0w_h0w_0198', b'1_kn0w_h0w_0199', b'1_kn0w_h0w_0200', b'1_kn0w_h0w_0201', b'1_kn0w_h0w_0202', b'1_kn0w_h0w_0203', b'1_kn0w_h0w_0204', b'1_kn0w_h0w_0205', b'1_kn0w_h0w_0206', b'1_kn0w_h0w_0207', b'1_kn0w_h0w_0208', b'1_kn0w_h0w_0209', b'1_kn0w_h0w_0210', b'1_kn0w_h0w_0211', b'1_kn0w_h0w_0212', b'1_kn0w_h0w_0213', b'1_kn0w_h0w_0214', b'1_kn0w_h0w_0215', b'1_kn0w_h0w_0216', b'1_kn0w_h0w_0217', b'1_kn0w_h0w_0218', b'1_kn0w_h0w_0219', b'1_kn0w_h0w_0220', b'1_kn0w_h0w_0221', b'1_kn0w_h0w_0222', b'1_kn0w_h0w_0223', b'1_kn0w_h0w_0224', b'1_kn0w_h0w_0225', b'1_kn0w_h0w_0226', b'1_kn0w_h0w_0227', b'1_kn0w_h0w_0228', b'1_kn0w_h0w_0229', b'1_kn0w_h0w_0230', b'1_kn0w_h0w_0231', b'1_kn0w_h0w_0232', b'1_kn0w_h0w_0233', b'1_kn0w_h0w_0234', b'1_kn0w_h0w_0235', b'1_kn0w_h0w_0236', b'1_kn0w_h0w_0237', b'1_kn0w_h0w_0238', b'1_kn0w_h0w_0239', b'1_kn0w_h0w_0240', b'1_kn0w_h0w_0241', b'1_kn0w_h0w_0242', b'1_kn0w_h0w_0243', b'1_kn0w_h0w_0244', b'1_kn0w_h0w_0245', b'1_kn0w_h0w_0246', b'1_kn0w_h0w_0247', b'1_kn0w_h0w_0248', b'1_kn0w_h0w_0249', b'1_kn0w_h0w_0250', b'1_kn0w_h0w_0251', b'1_kn0w_h0w_0252', b'1_kn0w_h0w_0253', b'1_kn0w_h0w_0254', b'1_kn0w_h0w_0255', b'1_kn0w_h0w_0256', b'1_kn0w_h0w_0257', b'1_kn0w_h0w_0258', b'1_kn0w_h0w_0259', b'1_kn0w_h0w_0260', b'1_kn0w_h0w_0261', b'1_kn0w_h0w_0262', b'1_kn0w_h0w_0263', b'1_kn0w_h0w_0264', b'1_kn0w_h0w_0265', b'1_kn0w_h0w_0266', b'1_kn0w_h0w_0267', b'1_kn0w_h0w_0268', b'1_kn0w_h0w_0269', b'1_kn0w_h0w_0270', b'1_kn0w_h0w_0271', b'1_kn0w_h0w_0272', b'1_kn0w_h0w_0273', b'1_kn0w_h0w_0274', b'1_kn0w_h0w_0275', b'1_kn0w_h0w_0276', b'1_kn0w_h0w_0277', b'1_kn0w_h0w_0278', b'1_kn0w_h0w_0279', b'1_kn0w_h0w_0280', b'1_kn0w_h0w_0281', b'1_kn0w_h0w_0282', b'1_kn0w_h0w_0283', b'1_kn0w_h0w_0284', b'1_kn0w_h0w_0285', b'1_kn0w_h0w_0286', b'1_kn0w_h0w_0287', b'1_kn0w_h0w_0288', b'1_kn0w_h0w_0289', b'1_kn0w_h0w_0290', b'1_kn0w_h0w_0291', b'1_kn0w_h0w_0292', b'1_kn0w_h0w_0293', b'1_kn0w_h0w_0294', b'1_kn0w_h0w_0295', b'1_kn0w_h0w_0296', b'1_kn0w_h0w_0297', b'1_kn0w_h0w_0298', b'1_kn0w_h0w_0299', b'1_kn0w_h0w_0300', b'1_kn0w_h0w_0301', b'1_kn0w_h0w_0302', b'1_kn0w_h0w_0303', b'1_kn0w_h0w_0304', b'1_kn0w_h0w_0305', b'1_kn0w_h0w_0306', b'1_kn0w_h0w_0307', b'1_kn0w_h0w_0308', b'1_kn0w_h0w_0309', b'1_kn0w_h0w_0310', b'1_kn0w_h0w_0311', b'1_kn0w_h0w_0312', b'1_kn0w_h0w_0313', b'1_kn0w_h0w_0314', b'1_kn0w_h0w_0315', b'1_kn0w_h0w_0316', b'1_kn0w_h0w_0317', b'1_kn0w_h0w_0318', b'1_kn0w_h0w_0319', b'1_kn0w_h0w_0320', b'1_kn0w_h0w_0321', b'1_kn0w_h0w_0322', b'1_kn0w_h0w_0323', b'1_kn0w_h0w_0324', b'1_kn0w_h0w_0325', b'1_kn0w_h0w_0326', b'1_kn0w_h0w_0327', b'1_kn0w_h0w_0328', b'1_kn0w_h0w_0329', b'1_kn0w_h0w_0330', b'1_kn0w_h0w_0331', b'1_kn0w_h0w_0332', b'1_kn0w_h0w_0333', b'1_kn0w_h0w_0334', b'1_kn0w_h0w_0335', b'1_kn0w_h0w_0336', b'1_kn0w_h0w_0337', b'1_kn0w_h0w_0338', b'1_kn0w_h0w_0339', b'1_kn0w_h0w_0340', b'1_kn0w_h0w_0341', b'1_kn0w_h0w_0342', b'1_kn0w_h0w_0343', b'1_kn0w_h0w_0344', b'1_kn0w_h0w_0345', b'1_kn0w_h0w_0346', b'1_kn0w_h0w_0347', b'1_kn0w_h0w_0348', b'1_kn0w_h0w_0349', b'1_kn0w_h0w_0350', b'1_kn0w_h0w_0351', b'1_kn0w_h0w_0352', b'1_kn0w_h0w_0353', b'1_kn0w_h0w_0354', b'1_kn0w_h0w_0355', b'1_kn0w_h0w_0356', b'1_kn0w_h0w_0357', b'1_kn0w_h0w_0358', b'1_kn0w_h0w_0359', b'1_kn0w_h0w_0360', b'1_kn0w_h0w_0361', b'1_kn0w_h0w_0362', b'1_kn0w_h0w_0363', b'1_kn0w_h0w_0364', b'1_kn0w_h0w_0365', b'1_kn0w_h0w_0366', b'1_kn0w_h0w_0367', b'1_kn0w_h0w_0368', b'1_kn0w_h0w_0369', b'1_kn0w_h0w_0370', b'1_kn0w_h0w_0371', b'1_kn0w_h0w_0372', b'1_kn0w_h0w_0373', b'1_kn0w_h0w_0374', b'1_kn0w_h0w_0375', b'1_kn0w_h0w_0376', b'1_kn0w_h0w_0377', b'1_kn0w_h0w_0378', b'1_kn0w_h0w_0379', b'1_kn0w_h0w_0380', b'1_kn0w_h0w_0381', b'1_kn0w_h0w_0382', b'1_kn0w_h0w_0383', b'1_kn0w_h0w_0384', b'1_kn0w_h0w_0385', b'1_kn0w_h0w_0386', b'1_kn0w_h0w_0387', b'1_kn0w_h0w_0388', b'1_kn0w_h0w_0389', b'1_kn0w_h0w_0390', b'1_kn0w_h0w_0391', b'1_kn0w_h0w_0392', b'1_kn0w_h0w_0393', b'1_kn0w_h0w_0394', b'1_kn0w_h0w_0395', b'1_kn0w_h0w_0396', b'1_kn0w_h0w_0397', b'1_kn0w_h0w_0398', b'1_kn0w_h0w_0399', b'1_kn0w_h0w_0400', b'1_kn0w_h0w_0401', b'1_kn0w_h0w_0402', b'1_kn0w_h0w_0403', b'1_kn0w_h0w_0404', b'1_kn0w_h0w_0405', b'1_kn0w_h0w_0406', b'1_kn0w_h0w_0407', b'1_kn0w_h0w_0408', b'1_kn0w_h0w_0409', b'1_kn0w_h0w_0410', b'1_kn0w_h0w_0411', b'1_kn0w_h0w_0412', b'1_kn0w_h0w_0413', b'1_kn0w_h0w_0414', b'1_kn0w_h0w_0415', b'1_kn0w_h0w_0416', b'1_kn0w_h0w_0417', b'1_kn0w_h0w_0418', b'1_kn0w_h0w_0419', b'1_kn0w_h0w_0420', b'1_kn0w_h0w_0421', b'1_kn0w_h0w_0422', b'1_kn0w_h0w_0423', b'1_kn0w_h0w_0424', b'1_kn0w_h0w_0425', b'1_kn0w_h0w_0426', b'1_kn0w_h0w_0427', b'1_kn0w_h0w_0428', b'1_kn0w_h0w_0429', b'1_kn0w_h0w_0430', b'1_kn0w_h0w_0431', b'1_kn0w_h0w_0432', b'1_kn0w_h0w_0433', b'1_kn0w_h0w_0434', b'1_kn0w_h0w_0435', b'1_kn0w_h0w_0436', b'1_kn0w_h0w_0437', b'1_kn0w_h0w_0438', b'1_kn0w_h0w_0439', b'1_kn0w_h0w_0440', b'1_kn0w_h0w_0441', b'1_kn0w_h0w_0442', b'1_kn0w_h0w_0443', b'1_kn0w_h0w_0444', b'1_kn0w_h0w_0445', b'1_kn0w_h0w_0446', b'1_kn0w_h0w_0447', b'1_kn0w_h0w_0448', b'1_kn0w_h0w_0449', b'1_kn0w_h0w_0450', b'1_kn0w_h0w_0451', b'1_kn0w_h0w_0452', b'1_kn0w_h0w_0453', b'1_kn0w_h0w_0454', b'1_kn0w_h0w_0455', b'1_kn0w_h0w_0456', b'1_kn0w_h0w_0457', b'1_kn0w_h0w_0458', b'1_kn0w_h0w_0459', b'1_kn0w_h0w_0460', b'1_kn0w_h0w_0461', b'1_kn0w_h0w_0462', b'1_kn0w_h0w_0463', b'1_kn0w_h0w_0464', b'1_kn0w_h0w_0465', b'1_kn0w_h0w_0466', b'1_kn0w_h0w_0467', b'1_kn0w_h0w_0468', b'1_kn0w_h0w_0469', b'1_kn0w_h0w_0470', b'1_kn0w_h0w_0471', b'1_kn0w_h0w_0472', b'1_kn0w_h0w_0473', b'1_kn0w_h0w_0474', b'1_kn0w_h0w_0475', b'1_kn0w_h0w_0476', b'1_kn0w_h0w_0477', b'1_kn0w_h0w_0478', b'1_kn0w_h0w_0479', b'1_kn0w_h0w_0480', b'1_kn0w_h0w_0481', b'1_kn0w_h0w_0482', b'1_kn0w_h0w_0483', b'1_kn0w_h0w_0484', b'1_kn0w_h0w_0485', b'1_kn0w_h0w_0486', b'1_kn0w_h0w_0487', b'1_kn0w_h0w_0488', b'1_kn0w_h0w_0489', b'1_kn0w_h0w_0490', b'1_kn0w_h0w_0491', b'1_kn0w_h0w_0492', b'1_kn0w_h0w_0493', b'1_kn0w_h0w_0494', b'1_kn0w_h0w_0495', b'1_kn0w_h0w_0496', b'1_kn0w_h0w_0497', b'1_kn0w_h0w_0498', b'1_kn0w_h0w_0499', b'1_kn0w_h0w_0500', b'1_kn0w_h0w_0501', b'1_kn0w_h0w_0502', b'1_kn0w_h0w_0503', b'1_kn0w_h0w_0504', b'1_kn0w_h0w_0505', b'1_kn0w_h0w_0506', b'1_kn0w_h0w_0507', b'1_kn0w_h0w_0508', b'1_kn0w_h0w_0509', b'1_kn0w_h0w_0510', b'1_kn0w_h0w_0511', b'1_kn0w_h0w_0512', b'1_kn0w_h0w_0513', b'1_kn0w_h0w_0514', b'1_kn0w_h0w_0515', b'1_kn0w_h0w_0516', b'1_kn0w_h0w_0517', b'1_kn0w_h0w_0518', b'1_kn0w_h0w_0519', b'1_kn0w_h0w_0520', b'1_kn0w_h0w_0521', b'1_kn0w_h0w_0522', b'1_kn0w_h0w_0523', b'1_kn0w_h0w_0524', b'1_kn0w_h0w_0525', b'1_kn0w_h0w_0526', b'1_kn0w_h0w_0527', b'1_kn0w_h0w_0528', b'1_kn0w_h0w_0529', b'1_kn0w_h0w_0530', b'1_kn0w_h0w_0531', b'1_kn0w_h0w_0532', b'1_kn0w_h0w_0533', b'1_kn0w_h0w_0534', b'1_kn0w_h0w_0535', b'1_kn0w_h0w_0536', b'1_kn0w_h0w_0537', b'1_kn0w_h0w_0538', b'1_kn0w_h0w_0539', b'1_kn0w_h0w_0540', b'1_kn0w_h0w_0541', b'1_kn0w_h0w_0542', b'1_kn0w_h0w_0543', b'1_kn0w_h0w_0544', b'1_kn0w_h0w_0545', b'1_kn0w_h0w_0546', b'1_kn0w_h0w_0547', b'1_kn0w_h0w_0548', b'1_kn0w_h0w_0549', b'1_kn0w_h0w_0550', b'1_kn0w_h0w_0551', b'1_kn0w_h0w_0552', b'1_kn0w_h0w_0553', b'1_kn0w_h0w_0554', b'1_kn0w_h0w_0555', b'1_kn0w_h0w_0556', b'1_kn0w_h0w_0557', b'1_kn0w_h0w_0558', b'1_kn0w_h0w_0559', b'1_kn0w_h0w_0560', b'1_kn0w_h0w_0561', b'1_kn0w_h0w_0562', b'1_kn0w_h0w_0563', b'1_kn0w_h0w_0564', b'1_kn0w_h0w_0565', b'1_kn0w_h0w_0566', b'1_kn0w_h0w_0567', b'1_kn0w_h0w_0568', b'1_kn0w_h0w_0569', b'1_kn0w_h0w_0570', b'1_kn0w_h0w_0571', b'1_kn0w_h0w_0572', b'1_kn0w_h0w_0573', b'1_kn0w_h0w_0574', b'1_kn0w_h0w_0575', b'1_kn0w_h0w_0576', b'1_kn0w_h0w_0577', b'1_kn0w_h0w_0578', b'1_kn0w_h0w_0579', b'1_kn0w_h0w_0580', b'1_kn0w_h0w_0581', b'1_kn0w_h0w_0582', b'1_kn0w_h0w_0583', b'1_kn0w_h0w_0584', b'1_kn0w_h0w_0585', b'1_kn0w_h0w_0586', b'1_kn0w_h0w_0587', b'1_kn0w_h0w_0588', b'1_kn0w_h0w_0589', b'1_kn0w_h0w_0590', b'1_kn
```

You have completed stage 1
You have completed stage 2
You have made it to the interim flag: cscg{y4y_1_50lv3d_7h3_f1r57_h4lf}
Bad input