Message From Space

Ivan Zaitsev

Jul 30, 2020

CONTENTS

This documentation contains artifacts from a joint deciphering effort of a mysterious radio transmission from outer space.

To skip the investigation part and go straight to the results, go to *Condensed Version*.

To read the whole story, go to *A Personal Appeal to Scientists and Engineers From All Over the World*. Beware: pages may contain speculations and unproven theories!

CHAPTER

ONE

A PERSONAL APPEAL TO SCIENTISTS AND ENGINEERS FROM ALL OVER THE WORLD

Hello.

My name is Ivan Zaitsev. I'm a staff astronomer at Pegovka observatory in the Urals region of Russia.

Several days ago we have been monitoring radio signals from a small region of sky around HD 190360 for our exoplanet research purposes. We have registered a peculiar radio transmission which cannot be attributed to any natural source in this area.

We tried to analyze and decode this message. But we don't have any trained deciphering specialists here on site, and we don't have appropriate software to analyze this message. We have made very little progress so far.

I believe that decoding this message can lead to a major breakthrough in our understanding of the Universe. I believe that science should be a joint effort. Together we can crack this problem much faster that any single research group.

That's why I decided to publish a `recording of this message` and create a *special documentation page* to collaborate on the explanation.

If you have any idea at all on how to decode and explain the message, please send a Pull Request to the appropriate page! We cannot move forward without this!

Sincerely, Ivan Zaitsev

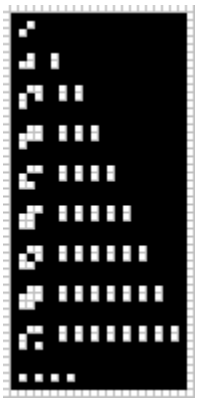
CHAPTER

TWO

CONDENSED VERSION

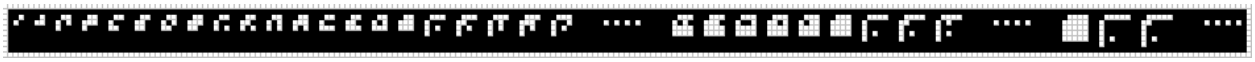
Note: Following documentation was generated automatically using a script contributed in our Discord chat. Decoded text representation may contain inaccuracies and will be revised with help from our participants.

2.1 #1. Numbers



```
0
1
2
3
4
5
6
7
8
...
```

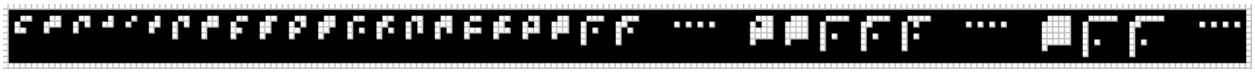
2.2 #2. Numbers (cont.)



Left vertical and top horizontal bars define glyph size. Number glyphs are always square. All other pixels except these bars are bits of encoded number written left to right and top to bottom. Top left pixel is always black in numbers.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
...
506 507 508 509 510 511 512 513 514
...
65535 65536 65537
...
```

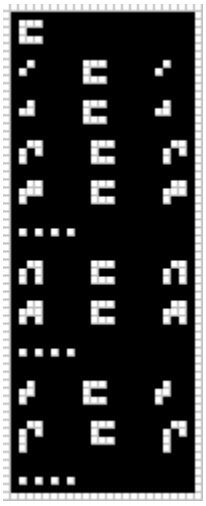
2.3 #3. Negative Numbers



Additional pixel in the vertical bar denotes a negative number.

```
4 3 2 1 0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10 -11 -12 -13 -14 -15 -16 -17
...
-510 -511 -512 -513 -514
...
-65535 -65536 -65537
...
```

2.4 #4. Equality



```
=
0   =   0
```

(continues on next page)

(continued from previous page)

```
1   =   1
2   =   2
3   =   3
...
10   =   10
11   =   11
...
-1   =   -1
-2   =   -2
...
```

2.5 #5. Successor

```
inc
ap inc 0   =   1
ap inc 1   =   2
ap inc 2   =   3
ap inc 3   =   4
...
ap inc 300   =   301
ap inc 301   =   302
...
ap inc -1   =   0
ap inc -2   =   -1
ap inc -3   =   -2
...
```

2.6 #6. Predecessor

```
dec
ap dec 1    =    0
ap dec 2    =    1
ap dec 3    =    2
ap dec 4    =    3
...
ap dec 1024    =    1023
...
ap dec 0    =    -1
ap dec -1    =    -2
ap dec -2    =    -3
...
```

2.7 #7. Sum

```
add
ap ap add 1 2    =    3
ap ap add 2 1    =    3
ap ap add 0 1    =    1
```

(continues on next page)

(continued from previous page)

```
ap ap add 2 3   =   5
ap ap add 3 5   =   8
...
```

2.8 #8. Variables

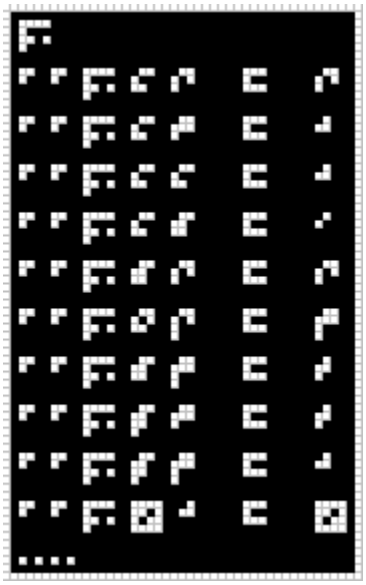
Variables have a white border. Inverted number inside the border identifies the variable.

```
x0 x1 x2 x3 x4   ...
ap ap add 0 x0   =   x0
ap ap add 0 x1   =   x1
ap ap add 0 x2   =   x2
...
ap ap add x0 0   =   x0
ap ap add x1 0   =   x1
ap ap add x2 0   =   x2
...
ap ap add x0 x1   =   ap ap add x1 x0
...
```

2.9 #9. Product

```
mul
ap ap mul 4 2    =    8
ap ap mul 3 4    =    12
ap ap mul 3 -2    =    -6
ap ap mul x0 x1    =    ap ap mul x1 x0
ap ap mul x0 0    =    0
ap ap mul x0 1    =    x0
...
```

2.10 #10. Integer Division



Rounds toward zero.

```
div
ap ap div 4 2    =    2
ap ap div 4 3    =    1
ap ap div 4 4    =    1
```

(continues on next page)

(continued from previous page)

```
ap ap div 4 5   =   0
ap ap div 5 2   =   2
ap ap div 6 -2   =   -3
ap ap div 5 -3   =   -1
ap ap div -5 3   =   -1
ap ap div -5 -3   =   1
ap ap div x0 1   =   x0
...
```

2.11 #11. Equality and Booleans

`t` is true, `f` is false.

```
eq
ap ap eq x0 x0   =   t
ap ap eq 0 -2    =   f
ap ap eq 0 -1    =   f
ap ap eq 0 0     =   t
ap ap eq 0 1     =   f
```

(continues on next page)

(continued from previous page)

```
ap ap eq 0 2   =   f
...
ap ap eq 1 -1   =   f
ap ap eq 1 0   =   f
ap ap eq 1 1   =   t
ap ap eq 1 2   =   f
ap ap eq 1 3   =   f
...
ap ap eq 2 0   =   f
ap ap eq 2 1   =   f
ap ap eq 2 2   =   t
ap ap eq 2 3   =   f
ap ap eq 2 4   =   f
...
ap ap eq 19 20   =   f
ap ap eq 20 20   =   t
ap ap eq 21 20   =   f
...
ap ap eq -19 -20   =   f
ap ap eq -20 -20   =   t
ap ap eq -21 -20   =   f
...
```

2.12 #12. Strict Less-Than

```
lt
ap ap lt 0 -1   =   f
ap ap lt 0 0    =   f
ap ap lt 0 1    =   t
ap ap lt 0 2    =   t
...
ap ap lt 1 0    =   f
ap ap lt 1 1    =   f
ap ap lt 1 2    =   t
ap ap lt 1 3    =   t
...
ap ap lt 2 1    =   f
ap ap lt 2 2    =   f
ap ap lt 2 3    =   t
ap ap lt 2 4    =   t
```

(continues on next page)

(continued from previous page)

```
...
ap ap lt 19 20   =   t
ap ap lt 20 20   =   f
ap ap lt 21 20   =   f
...
ap ap lt -19 -20   =   f
ap ap lt -20 -20   =   f
ap ap lt -21 -20   =   t
...
```

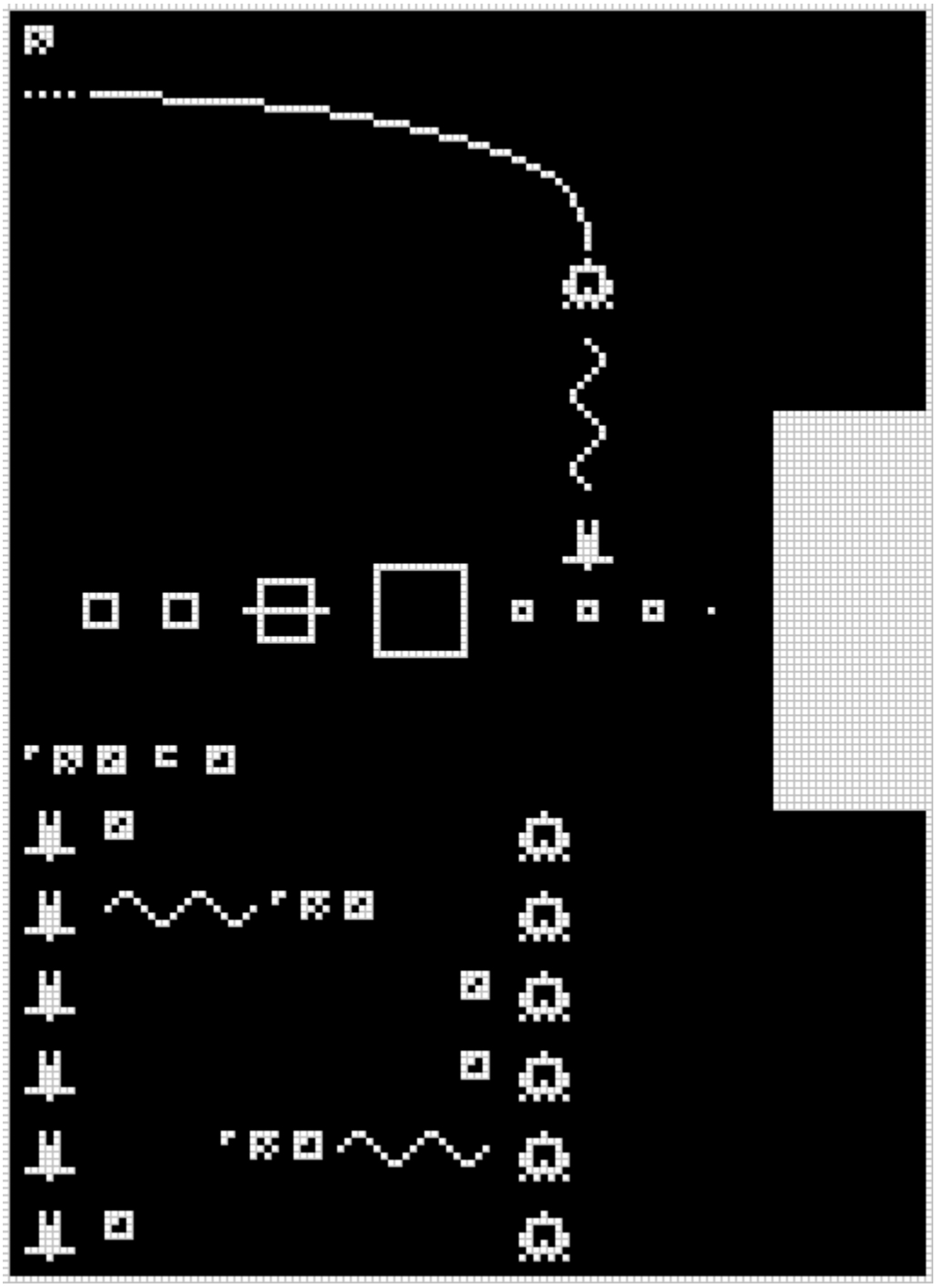
2.13 #13. Modulate

```
mod
ap mod 0   =   [0]
ap mod 1   =   [1]
ap mod -1   =   [-1]
ap mod 2   =   [2]
ap mod -2   =   [-2]
...
ap mod 16   =   [16]
ap mod -16   =   [-16]
...
ap mod 255   =   [255]
ap mod -255   =   [-255]
ap mod 256   =   [256]
ap mod -256   =   [-256]
...
```

2.14 #14. Demodulate

```
dem
ap dem ap mod x0    =    x0
ap mod ap dem x0    =    x0
```

2.15 #15. Send



```
send

ap send x0 =  x1

humans   x0                    aliens
humans   ~~~~~ ap mod x0       aliens
```

(continues on next page)

(continued from previous page)

```
humans                          x0 aliens
humans                          x1 aliens
humans           ap mod x1 ~~~~~ aliens
humans x1                        aliens
```

2.16 #16. Negate

```
neg
ap neg 0    =    0
ap neg 1    =    -1
ap neg -1    =    1
ap neg 2    =    -2
ap neg -2    =    2
...
```

2.17 #17. Function Application

`ap f x` is f(x)

The last two lines demonstrate that function application *ap* allows curried (i.e. partially evaluated) functions, by defining *inc* as the function *x* -> *1* + *x*.

```
ap
ap inc ap inc 0   =   2
ap inc ap inc ap inc 0   =   3
ap inc ap dec x0   =   x0
ap dec ap inc x0   =   x0
ap dec ap ap add x0 1   =   x0
ap ap add ap ap add 2 3 4   =   9
ap ap add 2 ap ap add 3 4   =   9
ap ap add ap ap mul 2 3 4   =   10
ap ap mul 2 ap ap add 3 4   =   14
inc   =   ap add 1
dec   =   ap add ap neg 1
...
```

2.18 #18. S Combinator

See https://en.wikipedia.org/wiki/B,_C,_K,_W_system

```
s
ap ap ap s x0 x1 x2   =   ap ap x0 x2 ap x1 x2
ap ap ap s add inc 1   =   3
ap ap ap s mul ap add 1 6   =   42
...
```

2.19 #19. C Combinator

See https://en.wikipedia.org/wiki/B,_C,_K,_W_system

```
c
ap ap ap c x0 x1 x2   =   ap ap x0 x2 x1
ap ap ap c add 1 2   =   3
...
```

2.20 #20. B Combinator

See https://en.wikipedia.org/wiki/B,_C,_K,_W_system

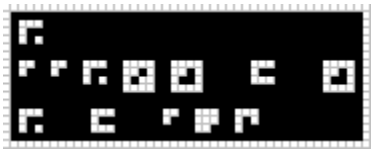
```
b
ap ap ap b x0 x1 x2   =   ap x0 ap x1 x2
ap ap ap b inc dec x0   =   x0
...
```

2.21 #21. True (K Combinator)

Decoded as `t`, because it has a meaning of boolean True.

```
t
ap ap t x0 x1   =   x0
ap ap t 1 5   =   1
ap ap t t i   =   t
ap ap t t ap inc 5   =   t
ap ap t ap inc 5 t   =   6
...
```

2.22 #22. False



Decoded as `f`, because it has a meaning of boolean False.

```
f
ap ap f x0 x1   =   x1
f   =   ap s t
```

2.23 #23. Power of Two

Recursive function: `pwr2` definition uses `pwr2`.

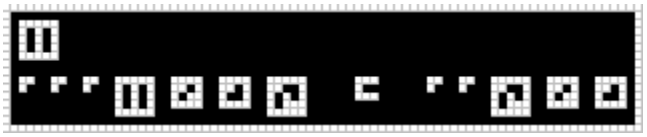
```
pwr2
pwr2   =   ap ap s ap ap c ap eq 0 1 ap ap b ap mul 2 ap ap b pwr2 ap add -1
ap pwr2 0   =   ap ap ap s ap ap c ap eq 0 1 ap ap b ap mul 2 ap ap b pwr2 ap add -1 0
ap pwr2 0   =   ap ap ap ap c ap eq 0 1 0 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 0
ap pwr2 0   =   ap ap ap ap eq 0 0 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 0
ap pwr2 0   =   ap ap t 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 0
ap pwr2 0   =   1
ap pwr2 1   =   ap ap ap s ap ap c ap eq 0 1 ap ap b ap mul 2 ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap ap ap c ap eq 0 1 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap ap ap eq 0 1 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap f 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap pwr2 ap ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap ap s ap ap c ap eq 0 1 ap ap b ap mul 2 ap ap b␣
↪pwr2 ap add -1 ap ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap ap ap c ap eq 0 1 ap ap add -1 1 ap ap ap b ap mul␣
↪2 ap ap b pwr2 ap add -1 ap ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap ap ap eq 0 ap ap add -1 1 1 ap ap ap b ap mul 2 ap␣
↪ap b pwr2 ap add -1 ap ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap ap ap eq 0 0 1 ap ap ap b ap mul 2 ap ap b pwr2 ap␣
↪add -1 ap ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap t 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 ap␣
↪ap add -1 1
ap pwr2 1   =   ap ap mul 2 1
ap pwr2 1   =   2
ap pwr2 2   =   ap ap ap s ap ap c ap eq 0 1 ap ap b ap mul 2 ap ap b pwr2 ap add -1 2
...
ap pwr2 2   =   4
ap pwr2 3   =   8
ap pwr2 4   =   16
ap pwr2 5   =   32
ap pwr2 6   =   64
ap pwr2 7   =   128
ap pwr2 8   =   256
...
```

2.24 #24. I Combinator

```
i(x) = x
```

```
i
ap i x0    =    x0
ap i 1    =    1
ap i i    =    i
ap i add    =    add
ap i ap add 1    =    ap add 1
...
```

2.25 #25. Cons (or Pair)



```
cons
ap ap ap cons x0 x1 x2    =    ap ap x2 x0 x1
```

2.26 #26. Car (First)

```
car
ap car ap ap cons x0 x1    =    x0
ap car x2    =    ap x2 t
```

2.27 #27. Cdr (Tail)

```
cdr
ap cdr ap ap cons x0 x1    =    x1
ap cdr x2    =    ap x2 f
```

2.28 #28. Nil (Empty List)

```
nil
ap nil x0   =   t
```

2.29 #29. Is Nil (Is Empty List)

```
isnil
ap isnil nil   =   t
ap isnil ap ap cons x0 x1   =   f
```

2.30 #30. List Construction Syntax

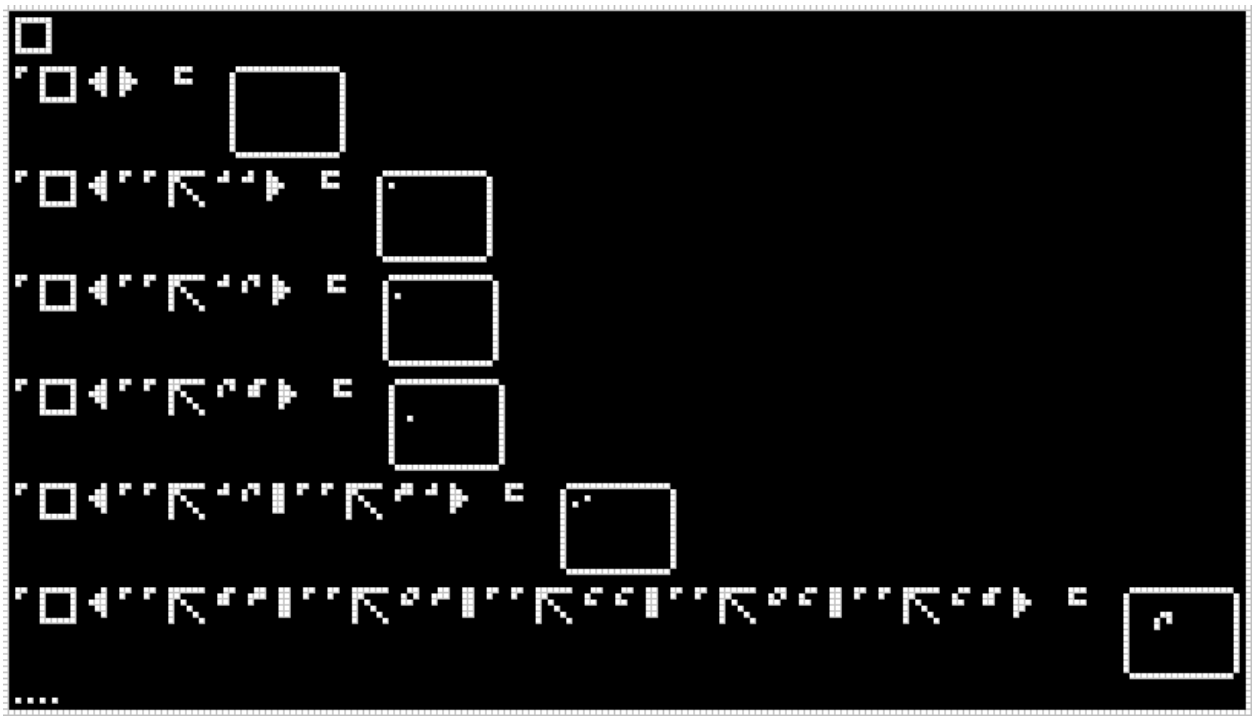
```
( , )
( )   =   nil
( x0 )   =   ap ap cons x0 nil
( x0 , x1 )   =   ap ap cons x0 ap ap cons x1 nil
( x0 , x1 , x2 )   =   ap ap cons x0 ap ap cons x1 ap ap cons x2 nil
( x0 , x1 , x2 , x5 )   =   ap ap cons x0 ap ap cons x1 ap ap cons x2 ap ap cons x5␣
↪nil
...
```

2.31 #31. Vector

Alias for `cons` that looks nice in "vector" usage context.

```
vec
vec   =   cons
```

2.32 #32. Draw



It draws a list of coordinates as dots on a picture.

```
draw
ap draw ( )   =   |picture1|
ap draw ( ap ap vec 1 1 )   =   |picture2|
ap draw ( ap ap vec 1 2 )   =   |picture3|
ap draw ( ap ap vec 2 5 )   =   |picture4|
ap draw ( ap ap vec 1 2 , ap ap vec 3 1 )   =   |picture5|
ap draw ( ap ap vec 5 3 , ap ap vec 6 3 , ap ap vec 4 4 , ap ap vec 6 4 , ap ap vec 4␣
↪5 )   =   |picture6|
...
```

2.33 #33. Checkerboard

Draws a checkerboard of the specified size.

```
checkerboard
checkerboard = ap ap s ap ap b s ap ap c ap ap b c ap ap b ap c ap c ap ap s ap ap b␣
↪s ap ap b ap b ap ap s i i lt eq ap ap s mul i nil ap ap s ap ap b s ap ap b ap b␣
↪cons ap ap s ap ap b s ap ap b ap b cons ap c div ap c ap ap s ap ap b b ap ap c ap␣
↪ap b b add neg ap ap b ap s mul div ap ap c ap ap b b checkerboard ap ap c add 2
ap ap checkerboard 7 0   =   |picture1|
ap ap checkerboard 13 0   =   |picture2|
```

2.34 #34. Multiple Draw

Takes a list of lists of 2D-points and returns a list of rendered pictures.

It applies draw function to all items of the list.

```
multipledraw
ap multipledraw nil   =   nil
ap multipledraw ap ap cons x0 x1   =   ap ap cons ap draw x0 ap multipledraw x1
```

2.35 #35. Modulate List

Apply *#13. Modulate* to a list constructed with *#25. Cons (or Pair)* or *#30. List Construction Syntax*.

```
mod cons
ap mod nil    =    [nil]
ap mod ap ap cons nil nil    =    [ap ap cons nil nil]
ap mod ap ap cons 0 nil    =    [ap ap cons 0 nil]
ap mod ap ap cons 1 2    =    [ap ap cons 1 2]
ap mod ap ap cons 1 ap ap cons 2 nil    =    [ap ap cons 1 ap ap cons 2 nil]
ap mod ( 1 , 2 )    =    [( 1 , 2 )]
ap mod ( 1 , ( 2 , 3 ) , 4 )    =    [( 1 , ( 2 , 3 ) , 4 )]
...
```

2.36 #36. Send (0)

:1678847 is decreasing over time at a rate of 1/3 per second and will reach 0 at the icfp contest main round deadline.

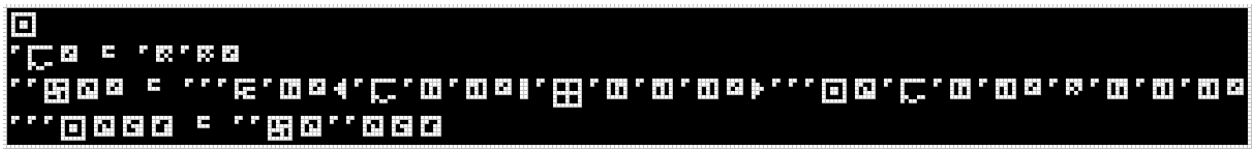
```
:1678847
ap send ( 0 )    =    ( 1 , :1678847 )
```

2.37 #37. Is 0

Function `if0` compares the first argument to 0 and picks the second argument if equal, else third.

```
if0
ap ap ap if0 0 x0 x1   =   x0
ap ap ap if0 1 x0 x1   =   x1
```

2.38 #38. Interact



Is a function that takes an "interaction-protocol", some data (maybe "protocol" dependent), and some pixel. It returns some new data, and a list of pictures.

Note that during the execution it sometimes uses the *send* function to communicate with spacecraft.

```
// list function call notation
f38 protocol (flag, newState, data) = if flag == 0
                then (modem newState, multipledraw data)
                else interact protocol (modem newState) (send data)
interact protocol state vector = f38 protocol (protocol state vector)



// mathematical function call notation
f38(protocol, (flag, newState, data)) = if flag == 0
                then (modem(newState), multipledraw(data))
                else interact(protocol, modem(newState), send(data))
interact(protocol, state, vector) = f38(protocol, protocol(state, vector))
```

`mod` is defined on `cons`, `nil` and `numbers` only. So `modem` function seems to be the way to say that it's argument consists of numbers and lists only.

So we can assume that `newState` is always list of list of ... of numbers.

After experimenting with the galaxy interaction protocol we have found out several good ideas:

1. We can choose any `vector` to pass it to the interact function. But a convenient way to input this vectors — is clicking on the image pixel from the previous interact execution result.
3. We need to draw the images passed to multipledraw somehow. A convenient way to do it — is to overlay images one over another using different colors for different images.

```
interact
ap modem x0 = ap dem ap mod x0
ap ap f38 x2 x0 = ap ap ap if0 ap car x0 ( ap modem ap car ap cdr x0 , ap␣
↪multipledraw ap car ap cdr ap cdr x0 ) ap ap ap interact x2 ap modem ap car ap cdr␣
↪x0 ap send ap car ap cdr ap cdr x0
ap ap ap interact x2 x4 x3 = ap ap f38 x2 ap ap x2 x4 x3
```

2.39 #39. Interaction Protocol

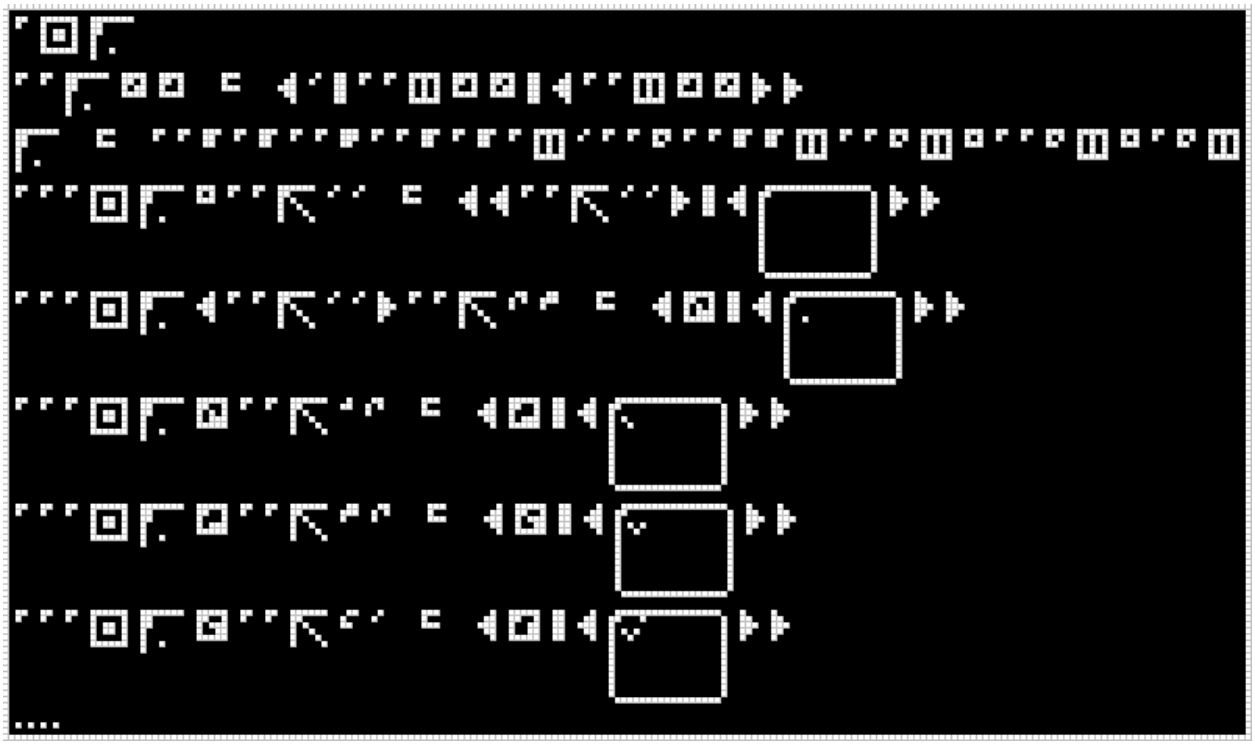
Start the protocol passing `nil` as the initial state and `(0, 0)` as the initial point. Then iterate the protocol passing new points along with states obtained from the previous execution.

```
interact
ap ap ap interact x0 nil ap ap vec 0 0 = ( x16 , ap multipledraw x64 )
ap ap ap interact x0 x16 ap ap vec x1 x2 = ( x17 , ap multipledraw x65 )
ap ap ap interact x0 x17 ap ap vec x3 x4 = ( x18 , ap multipledraw x66 )
ap ap ap interact x0 x18 ap ap vec x5 x6 = ( x19 , ap multipledraw x67 )
...
```

2.40 #40. Stateless Drawing Protocol

```
ap interact statelessdraw
ap ap statelessdraw nil x1 = ( 0 , nil , ( ( x1 ) ) )
statelessdraw = ap ap c ap ap b b ap ap b ap b ap cons 0 ap ap c ap ap b b cons ap ap␣
↪c cons nil ap ap c ap ap b cons ap ap c cons nil nil
ap ap ap interact statelessdraw nil ap ap vec 1 0 = ( nil , ( [1,0] ) )
ap ap ap interact statelessdraw nil ap ap vec 2 3 = ( nil , ( [2,3] ) )
ap ap ap interact statelessdraw nil ap ap vec 4 1 = ( nil , ( [4,1] ) )
...
```

2.41 #41. Stateful Drawing Protocol



It gives us back the variable bound to the draw state, so we can set the next pixel with the next call.

```
ap interact :67108929
ap ap :67108929 x0 x1 = ( 0 , ap ap cons x1 x0 , ( ap ap cons x1 x0 ) )
:67108929 = ap ap b ap b ap ap s ap ap b ap b ap cons 0 ap ap c ap ap b b cons ap ap␣
↪c cons nil ap ap c cons nil ap c cons
ap ap ap interact :67108929 nil ap ap vec 0 0 = ( ( ap ap vec 0 0 ) , ( [0,0] ) )
ap ap ap interact :67108929 ( ap ap vec 0 0 ) ap ap vec 2 3 = ( x2 , ( [0,0;2,3] ) )
ap ap ap interact :67108929 x2 ap ap vec 1 2 = ( x3 , ( [0,0;2,3;1,2] ) )
ap ap ap interact :67108929 x3 ap ap vec 3 2 = ( x4 , ( [0,0;2,3;1,2;3,2] ) )
ap ap ap interact :67108929 x4 ap ap vec 4 0 = ( x5 , ( [0,0;2,3;1,2;3,2;4,0] ) )
...
```

2.42 #42. Galaxy

We believe that this message tells us to run an *interaction protocol* called `galaxy`. This protocol is defined in the last line of the `huge message` included on this page.

Messages *#38* and *#39* describe how to run a protocol. As we can see from *message #38*, a protocol takes a *vector* and returns a list of *pictures*.

Messages *#40* and *#41* define two simple protocols and demonstrate their behavior during execution.

```
ap interact galaxy = ...
```

CHAPTER

THREE

#1. NUMBERS

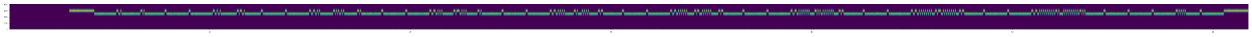
Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

`Download` radio transmission recording. It was originally received at ~5 GHz and scaled down to ~500 Hz to make signal audible for humans.

3.1 Spectrogram

Spectrogram of the recording, rendered with a notebook by Discord user @nya:

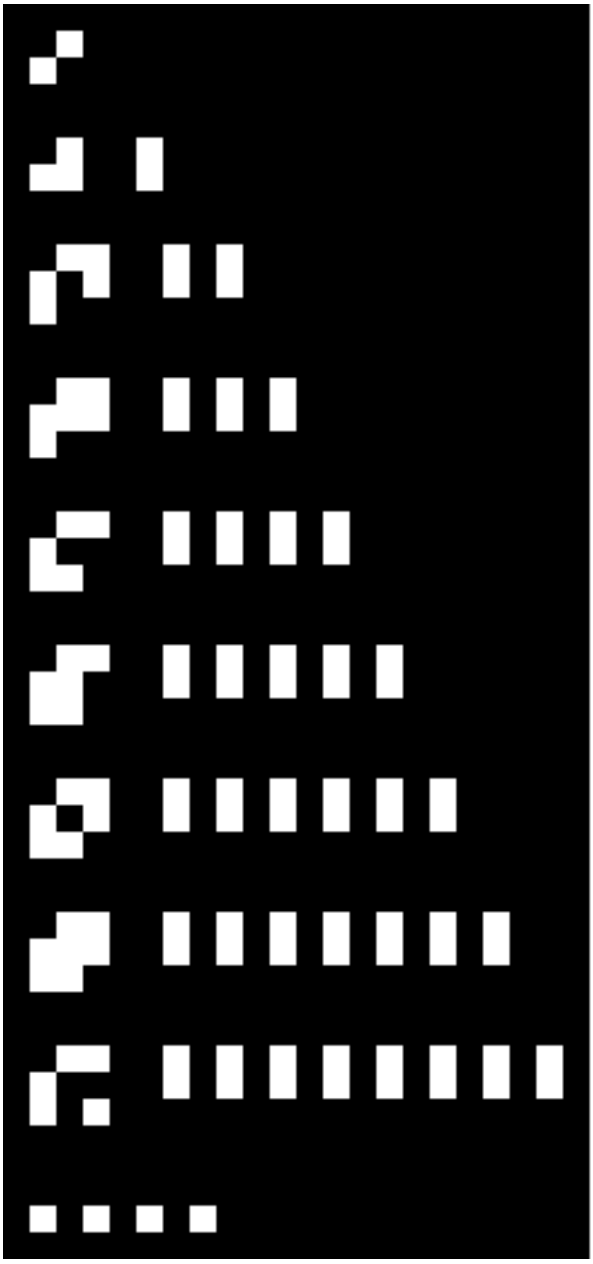


3.2 Image

A 2D image created by:

1. Converting low and high frequency spans into black and white squares respectively.
2. Rearranging these squares into a rectangle instead of a single line.

Contributed by Discord user @elventian.

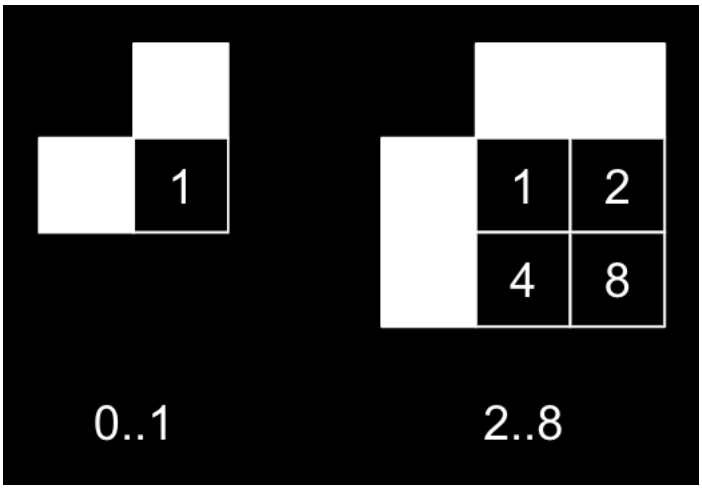


3.3 Interpretation

Based on the discussions with Discord users @nya, @Kilew, @fryguybob, @aaaa1, @gltronred and @elventian.

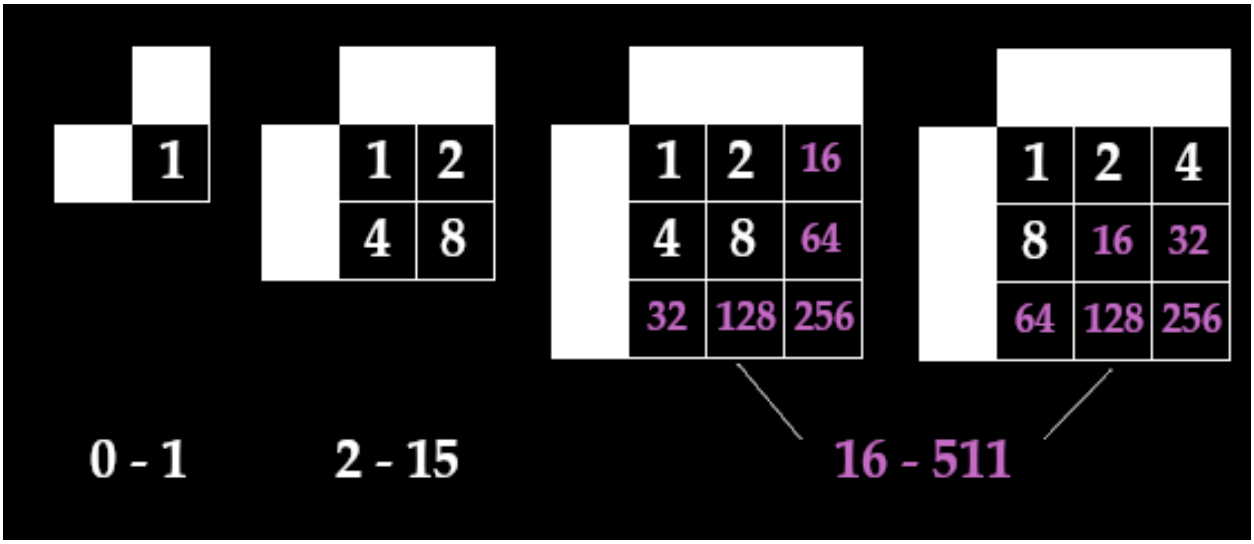
Probably the symbols on the left represent numbers and the number of elements on the right is the unary representation of this number.

Symbols on the left look like a binary encoding that should work for numbers 1..15. Picture says 8, because we have hard data only up to 8:



According to this theory we can speculate that the numbers 9..15 would be represented with these symbols:

Based on this logic the symbols could be extended further like this:



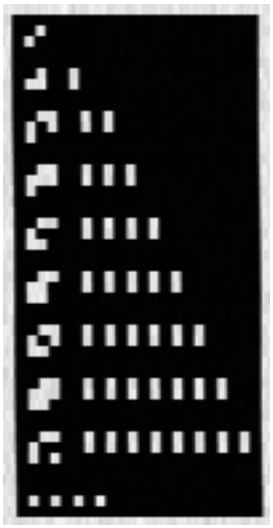
Second transmission seems to support the second conjecture (right picture).

3.4 Code

This Rust code generates decoded images similar to the image included above from WAV files.

Contributed by Discord user @aaaa1.

Example output:



```rust
// [dependencies]
// png = "0.16"
// hound = "3.4"

fn main() {
    let r = hound::WavReader::open("radio-transmission-recording.wav").unwrap();
    let spec = r.spec();
    let samples: Vec<i16> = r.into_samples().map(Result::unwrap).collect();

    let freq = 600;
    let step = 2.0 * std::f32::consts::PI * freq as f32 / spec.sample_rate as f32;
    let xys: Vec<(f32, f32)> = samples.iter().copied().enumerate().map(|(i, s)| {
        let s = s as f32;
        let a = i as f32 * step;
        (a.cos() * s, a.sin() * s)
    }).collect();

    let mut axyz = vec![(0.0, 0.0)];
    for (x, y) in xys {
        let last = *axyz.last().unwrap();
        axyz.push((last.0 + x, last.1 + y));
    }

    let mut ds: Vec<f32> = axyz.iter().zip(axyz.iter().skip(1000)).map(|(xy1, xy2)| {
        let dx = xy1.0 - xy2.0;
        let dy = xy1.1 - xy2.1;
        dx * dx + dy * dy
    }).collect();
    let max = *ds.iter().max_by(|x, y| x.partial_cmp(y).unwrap()).unwrap();
    ds.iter_mut().for_each(|x| *x /= max);
```

(continues on next page)

(continued from previous page)

```
    let width = 100usize;
    let height = 195usize;

    let w = std::fs::File::create("res.png").unwrap();
    let w = std::io::BufWriter::new(w);
    let mut encoder = png::Encoder::new(w, width as u32, height as u32);
    encoder.set_color(png::ColorType::Grayscale);
    encoder.set_depth(png::BitDepth::Eight);
    let mut w = encoder.write_header().unwrap();

    let mut data = vec![0u8; width * height];
    for (i, cell) in data.iter_mut().enumerate() {
        let x = i % width;
        let y = i / width / 4;
        *cell = (ds.get((x + y * width) * 529 + 132400).copied().unwrap_or(0.0) * 255.
↪0) as u8;
    }
    w.write_image_data(&data).unwrap();
}
```

CHAPTER

FOUR

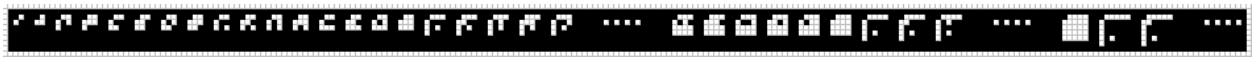
#2. NUMBERS (CONT.)

Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

4.1 Image

This image was produced from the second radio transmission using *previously contributed code*.



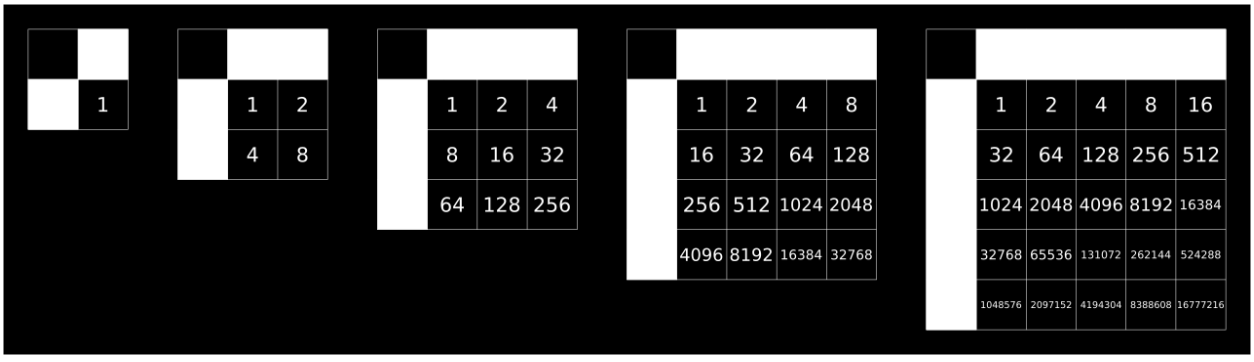
4.2 Interpretation

Contributed by Discord user @elventian.

We have enough data to conclude that we've found the way of encoding natural numbers by raster monochrome pictogram framed at the top and left. There is a square semantic region with side N inside the pictogram. Each pixel of the region corresponds to one bit in the binary notation of the number. Let x and y be column and row numbers in the range [0 . . . N), then the place value for the cell (x, y) is determined by the following formula:

$$place\_value(x, y) = 2^{y*N+x}$$

Place values for N up to 5:



4.3 Decoded

Decoded by Discord users @gltronred and @frictionless.

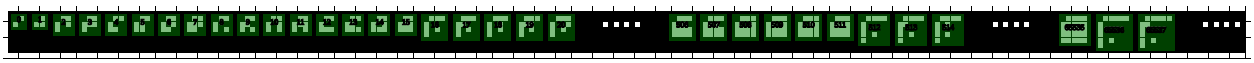
```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
...
506 507 508 509 510 511 512 513 514
...
65535 65536 65537
...
```

4.4 Code

This Python code decodes and annotates numbers on a provided picture.

Contributed by Discord user @pink_snow.

Example output:



```
#!/usr/bin/env python
#! nix-shell -i python -p "python38.withPackages(p:[p.pillow])"

import sys
from PIL import Image


class Img:
    def __init__(self, fname, zoom):
        self._img = Image.open(fname)
        self._pixels = self._img.load()
        self._zoom = zoom

        self.size = self._img.size[0] // zoom, self._img.size[1] // zoom

    def __getitem__(self, xy):
        xy = xy[0] * self._zoom, xy[1] * self._zoom
        try:
            c = self._pixels[xy]
        except IndexError:
            return False
        return c[0] + c[1] + c[2] > 382

    def dump(self, ix, iy, higlight = set()):
        for y in iy:
            for x in ix:
                if (x,y) in higlight:
                    print(end="\x1b[40;31m")
                print(end=".#"[self[x,y]])
                if (x,y) in higlight:
                    print(end="\x1b[m")
            print()
        print()
```

(continues on next page)

(continued from previous page)

```
class Svg:
    def __init__(self, fname, width, height):
        self._f = open(fname, "w")
        self._print(
            f'<svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="{width*8}"␣
↪height="{height*8}">'
        )
        self._print(f'<rect width="{width*8}" height="{height*8}" style="fill:black"/>
↪')

    def _print(self, *args, **kwargs):
        print(*args, **kwargs, file=self._f)

    def point(self, x, y):
        self._print(
            f'<rect x="{x*8}" y="{y*8}" width="7" height="7" style="fill:white"/>'
        )

    def annotation(self, x, y, w, h, text):
        self._print(
            f'<rect x="{x*8}" y="{y*8}" width="{w*8}" height="{h*8}" style=
↪"fill:green;opacity:0.5"/>'
        )
        style = "paint-order: stroke; fill: white; stroke: black; stroke-width: 2px;␣
↪font:24px bold sans;"
        self._print(
            f'<text x="{x*8+w*4}" y="{y*8+h*4}" dominant-baseline="middle" text-
↪anchor="middle" fill="white" style="{style}">{text}</text>'
        )

    def close(self):
        self._print("</svg>")
        self._f.close()


def decode_number(img, x, y):
    if img[x - 1, y - 1] or img[x, y - 1] or img[x - 1, y] or img[x, y]:
        return None

    # Get the size by iterating over top and left edges
    size = 0
    negative = False
    while True:
        items = (
            img[x + size + 1, y - 1],
            img[x + size + 1, y],
            img[x - 1, y + size + 1],
            img[x, y + size + 1],
        )
        if items == (False, True, False, True):
            size += 1
            continue
        if items == (False, False, False, False):
            break
        if items == (False, False, False, True):
            negative = True
```

(continues on next page)

(continued from previous page)

```
            break
        return None

    if size == 0:
        return None

    # Check that right and bottom edges are empty
    for i in range(1,size + 2):
        if img[x + size + 1, y+i] or img[x+i, y + size + 1]:
            return None

    # Decode the number
    result, d = 0, 1
    for iy in range(size):
        for ix in range(size):
            result += d * img[x + ix + 1, y + iy + 1]
            d *= 2

    if negative:
        result = -result

    return (size, size+negative), result


def main(in_fname, out_fname):
    img = Img(in_fname, 4)
    svg = Svg(out_fname, img.size[0], img.size[1])

    for y in range(img.size[1]):
        for x in range(img.size[0]):
            if img[x, y]:
                svg.point(x, y)

    for y in range(img.size[1]):
        for x in range(img.size[0]):
            if (n := decode_number(img, x, y)) is not None:
                svg.annotation(x - 0.5, y - 0.5, n[0][0] + 2, n[0][1] + 2, n[1])
    svg.close()


main(sys.argv[1], sys.argv[2])
```

CHAPTER

FIVE

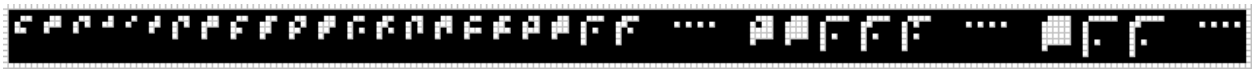
#3. NEGATIVE NUMBERS

Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

5.1 Image

This image was produced from the third radio transmission using *previously contributed code*.



This partly annotated version of the image was made using *code from message #2*.

5.2 Interpretation

Contributed by Discord user @pink_snow.

Looks like the bottom left additional pixel is used to indicate negative numbers.

5.3 Decoded

```
4 3 2 1 0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10 -11 -12 -13 -14 -15 -16 -17
...
-510 -511 -512 -513 -514
...
-65535 -65536 -65537
...
```

5.4 Code

Revised version of the Python code that supports negative numbers is published on the *message #2 page*.

Contributed by Discord user @pink_snow.

Example output:

@pink_snow also provided a Haskell version of the same code.

```
#! /usr/bin/env nix-shell
-- Usage: ./annotate.hs in-msg.png out-annotated.svg out-decoded.txt
#! nix-shell -i runhaskell -p
#! nix-shell "haskellPackages.ghcWithPackages (pkgs: with pkgs; [JuicyPixels␣
↪JuicyPixels-util errors extra groupBy])"
#! nix-shell -I nixpkgs=https://github.com/NixOS/nixpkgs/archive/
↪5cb5ccb54229efd9a4cd1fccc0f43e0bbed81c5d.tar.gz

import Control.Arrow ((&&&))
import Control.Error.Safe (justZ)
import Control.Monad (forM, forM_, guard)
import Control.Monad.ST (runST)
import Control.Monad.Trans (lift)
import Control.Monad.Trans.Maybe (runMaybeT)
import Data.List (foldl', intercalate, sortOn)
import Data.List.Extra (replace)
import Data.List.GroupBy (groupBy)
import Data.Maybe (catMaybes, fromMaybe)
import Data.Word (Word16)
import Numeric (showHex)
import System.Environment (getArgs)
import qualified Codec.Picture as P
import qualified Codec.Picture.RGBA8 as P8
import qualified Data.Vector.Mutable as V

--------------------------------------------------------------------------------
-- Types

type Scale = Int
type Coord = (Int, Int)
type Size = (Int, Int)

data Img = Img (P.Image P.PixelRGBA8) Scale

data Symbol
  = SymNumber Integer
  | SymModulatedNumber Integer
  | SymOperator Integer
  | SymVariable Integer
  | SymEllipsis
  | SymSpecial String
  | SymBox Int Int Integer
  | SymUnknown

--------------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```haskell
-- Misc functions

range2d :: Int -> Int -> Int -> Int -> [(Int, Int)]
range2d x0 y0 x1 y1 = [(x', y') | y' <- [y0 .. y1], x' <- [x0 .. x1]]

none :: (a -> Bool) -> [a] -> Bool
none = (not .) . any

bitsToInteger :: [Bool] -> Integer
bitsToInteger = fst . foldl' f (0, 1)
  where
    f (sum, bit) True = (sum + bit, bit*2)
    f (sum, bit) False = (sum, bit*2)

groupAcc :: (a -> s) -> (s -> a -> Maybe s) -> [a] -> [(s, [a])]
groupAcc init f = groupAcc1'
  where
    groupAcc1' [] = []
    groupAcc1' (x:xs) = takeGroup (init x) [x] xs

    takeGroup state group [] = [(state, reverse group)]
    takeGroup state group (y:ys) = case f state y of
      Nothing -> (state, reverse group) : groupAcc1' (y:ys)
      Just state' -> takeGroup state' (y : group) ys

-------------------------------------------------------------------------------
-- Img

imgLoad :: FilePath -> Scale -> IO Img
imgLoad path scale = (\img -> Img img scale) <$> P8.readImageRGBA8 path

imgWidth :: Img -> Int
imgWidth (Img img scale) = P.imageWidth img `div` scale

imgHeight :: Img -> Int
imgHeight (Img img scale) = P.imageHeight img `div` scale

imgPixel :: Img -> Coord -> Bool
imgPixel (Img img scale) (x, y) = True
  && x' >= 0 && y' >= 0
  && x' < P.imageWidth img && y' < P.imageHeight img
  && fromIntegral r + fromIntegral g + fromIntegral b > (0::Word16)
  where
    (x', y') = (x * scale, y * scale)
    P.PixelRGBA8 r g b _ = P.pixelAt img x' y'

imgShow :: Img -> [Int] -> [Int] -> String
imgShow img xs ys = unlines $ map showLine ys
  where showLine y = map (\x -> if imgPixel img (x, y) then '#' else '.') xs

imgShowFull :: Img -> String
imgShowFull img = imgShow img [0 .. imgWidth img - 1] [0 .. imgHeight img - 1]

instance Show Img where
  show = imgShowFull

imgAllPixels :: Img -> [Coord]
```

(continues on next page)

(continued from previous page)

```haskell
imgAllPixels img = range2d 0 0 (imgWidth img - 1) (imgHeight img - 1)

--------------------------------------------------------------------------------
-- Symbol decoder

symDecode :: Img -> Coord -> Size -> Symbol
symDecode img (x, y) (w, h)
  | checkSymbol img symGalaxy (x, y) = SymSpecial "galaxy"
  | checkSymbol img symHuman (x, y) = SymSpecial "human"
  | checkSymbol img symSpacecraft (x, y) = SymSpecial "spacecraft"
  | isNonNegativeNumber = SymNumber value
  | isNegativeNumber = SymNumber (-value)
  | isModulatedNumber = SymModulatedNumber modulatedValue
  | isVariable = SymVariable varValue
  | isOperator = SymOperator value
  | isBox = SymBox (w-2) (h-2) boxValue
  | isEllipsis = SymEllipsis
  | checkSymbol img symOpenPar (x-1, y-1) = SymSpecial "("
  | checkSymbol img symClosePar (x-1, y-1) = SymSpecial ")"
  | checkSymbol img symPipe (x-1, y-1) = SymSpecial ","
  | otherwise = SymUnknown
  where
    size = w

    px (x', y') = imgPixel img (x + x', y + y')

    isNonNegativeNumber = True
      && w == h
      && not (px (0, 0))

    isNegativeNumber = True
      && size >= 2
      && w + 1 == h
      && not (px (0, 0))
      && px (0, size)
      && none px [(x', size) | x' <- [1 .. size-1]] -- bottom + 1 is empty

    isOperator = True
      && w == h
      && px (0, 0)

    isVariable = True
      && size >= 4
      && w == h
      && size >= 4
      && px (1, 1)
      && all px [(x',      size-1) | x' <- [0 .. size-1]] -- bottom is full
      && all px [(size-1, y')      | y' <- [0 .. size-1]] -- right is full
      && none px [(x', 1) | x' <- [2 .. size-2]] -- top + 1 is empty
      && none px [(1, y') | y' <- [2 .. size-2]] -- left + 1 is empty

    isBox = True
      && not (px (0,   0))
      && not (px (w-1, 0))
      && not (px (0,   h-1))
      && not (px (w-1, h-1))
      && all px [(x',  h-1) | x' <- [1 .. w-2]] -- bottom is full
```

(continues on next page)

(continued from previous page)

```
      && all px [(w-1, y')  | y' <- [1 .. h-2]] -- right is full

    isEllipsis = checkSymbol img symEllipsis (x-1, y-1)

    isModulatedNonNeg = checkSymbol img symModulatedNonNeg (x, y)

    isModulatedNeg = checkSymbol img symModulatedNeg (x, y)

    isModulatedNumber = isModulatedNonNeg || isModulatedNeg

    value = bitsToInteger $ map px $ range2d 1 1 (size-1) (size-1)

    varValue = bitsToInteger $ map (not . px) $ range2d 2 2 (size-2) (size-2)

    boxValue = bitsToInteger $ map px $ range2d 1 1 (w-2) (h-2)

    modulatedSign = if isModulatedNonNeg then 1 else (-1)

    modulatedNibbleCount = length $ takeWhile (\x -> px (x, 0)) [2 ..]

    modulatedBits = map (\x -> px (x, 0)) $
      take (4 * modulatedNibbleCount) [(3 + modulatedNibbleCount) ..]

    modulatedValue = modulatedSign * (foldl (\acc bit -> (2 * acc) + if bit then 1␣
↪else 0) 0 modulatedBits)

symDetectAll :: Img -> [(Coord, Size)]
symDetectAll img = runST $ do
  vec <- V.replicate (width * height) False
  fmap catMaybes $ forM validRange $ \(x, y) -> runMaybeT $ do
    guard =<< not <$> V.read vec (idx (x, y))
    (w, h) <- justZ $ symDetectSingle img (x, y)
    lift $ forM_ (range2d x y (x+w-1) (y+h-1)) $ flip (V.write vec) True . idx
    return ((x, y), (w, h))
  where
    (width, height) = (imgWidth &&& imgHeight) img
    validRange = range2d 2 2 (width - 3) (height - 3)
    idx (x, y) = x + y * width

symDetectSingle :: Img -> Coord -> Maybe Size
symDetectSingle img (x, y)
  | checkSymbol img symModulatedNonNeg (x, y) = Just (modulatedWidth, 2)
  | checkSymbol img symModulatedNeg (x, y) = Just (modulatedWidth, 2)
  | px 1 0 && px 0 1 && nothingBelow = Just (gridWidth + 1, gridHeight + 1)
  | not (px 0 0) && px 1 0 && px 0 1 && isBox = Just (gridWidth + 2, gridHeight + 2)
  | checkSymbol img symEllipsis (x-1, y-1) = Just (7, 1)
  | checkSymbol img symOpenPar (x-1, y-1) = Just (3, 5)
  | checkSymbol img symClosePar (x-1, y-1) = Just (3, 5)
  | checkSymbol img symPipe (x-1, y-1) = Just (2, 5)
  | checkSymbol img symGalaxy (x, y) = Just (7, 7)
  | checkSymbol img symHuman (x, y) = Just (7, 7)
  | checkSymbol img symSpacecraft (x, y) = Just (7, 7)
  | otherwise = Nothing
  where
    px x' y' = imgPixel img (x + x', y + y')
    gridWidth = length $ takeWhile (flip px 0) [1 ..]
    gridHeight = length $ takeWhile (px 0) [1 ..]
```

(continues on next page)

(continued from previous page)

```
    modulatedWidth = length $ takeWhile (\x -> px x 0 || px x 1) [0 ..]
    nothingBelow = none (\x -> px x (gridHeight+1)) [0..gridWidth]
    isBox = all (\i -> px (gridWidth+1) i) [1..gridHeight-1]

symEllipsis :: [[Bool]]
symEllipsis = map (map (=='#'))
  [ "........."
  , ".#.#.#.#."
  , "........."
  ]

symModulatedNonNeg :: [[Bool]]
symModulatedNonNeg = map (map (=='#'))
  [ ".#"
  , "#."
  , "."
  ]

symModulatedNeg :: [[Bool]]
symModulatedNeg = map (map (=='#'))
  [ "#."
  , ".#"
  ]

symOpenPar :: [[Bool]]
symOpenPar = map (map (=='#'))
  [ "....."
  , "...#."
  , "..##."
  , ".###."
  , "..##."
  , "...#."
  , "....."
  ]

symClosePar :: [[Bool]]
symClosePar = map (map (=='#'))
  [ "....."
  , ".#.."
  , ".##."
  , ".###"
  , ".##."
  , ".#.."
  , "...."
  ]

symPipe :: [[Bool]]
symPipe = map (map (=='#'))
  [ "...."
  , ".##."
  , ".##."
  , ".##."
  , ".##."
  , ".##."
  , "...."
  ]
```

(continues on next page)

(continued from previous page)

```
symGalaxy :: [[Bool]]
symGalaxy = map (map (=='#'))
  [ "..###.."
  , ".....#."
  , ".###..#"
  , "#.#.#.#"
  , "#..###."
  , " #....."
  , "..###.."
  ]

symHuman :: [[Bool]]
symHuman = map (map (=='#'))
  [ "..#.#.."
  , "..#.#.."
  , "..###.."
  , "..###.."
  , "..###.."
  , "#######"
  , "...#..."
  ]

symSpacecraft :: [[Bool]]
symSpacecraft = map (map (=='#'))
  [ "...#..."
  , ".#####."
  , ".#...#."
  , "##...##"
  , "##.#.##"
  , ".#####."
  , "#.#.#.#"
  ]

checkLine :: Img -> [Bool] -> Coord -> Bool
checkLine _ [] _ = True
checkLine img (h:t) (x, y) = (imgPixel img (x, y) == h) && checkLine img t (x+1, y)

checkSymbol :: Img -> [[Bool]] -> Coord -> Bool
checkSymbol _ [] _ = True
checkSymbol img (h:t) (x, y) = checkLine img h (x, y) && checkSymbol img t (x, y+1)

--------------------------------------------------------------------------------
-- svg

svg :: Img -> [(Coord, Size, String, String)] -> String
svg img annotations =
  concat (
    svgHead img ++
    svgImgPoints img ++
    concatMap (\(coord, size, text, color) -> svgAnnotation coord size text color)␣
↪annotations ++
    ["</svg>"]
  )

svgHead :: Img -> [String]
svgHead img = [
    "<svg xmlns='http://www.w3.org/2000/svg' version='1.1' width='",
```

(continues on next page)

(continued from previous page)

```
    show $ 1 + imgWidth img * 8,
    "' height='",
    show $ 1 + imgHeight img * 8,
    "'>\n",
    "<rect width='100%' height='100%' style='fill:black'/>\n"
  ]

svgPoint :: Coord -> Bool -> [String]
svgPoint (x, y) value = [
    "<rect x='", show (1 + x*8),
    "' y='", show (1 + y*8),
    "' width='7' height='7' style='fill:",
    if value then "white" else "#333333",
    "'/>\n"
  ]

svgImgPoints :: Img -> [String]
svgImgPoints img =
  concatMap (\coord -> svgPoint coord (imgPixel img coord)) $ imgAllPixels img

svgAnnotation :: Coord -> Size -> String -> String -> [String]
svgAnnotation (x, y) (w, h) text color = [
    "<rect x='", show (1 + x*8 - 6),
    "' y='", show (1 + y*8 - 6),
    "' width='", show (w*8 + 11),
    "' height='", show (h*8 + 11),
    "' style='fill:", color, ";opacity:0.5'/>\n",

    "<text x='", show (1 + x*8 + w*4),
    "' y='", show (1 + y*8 + h*4),
    "' dominant-baseline='middle' text-anchor='middle' fill='white' style='",
    "paint-order: stroke; fill: white; stroke: black; stroke-width: 2px; font:24px␣
↪bold sans;",
    "'>", text', "</text>\n"
  ]
  where text' = replace "<" "&lt;" text

-------------------------------------------------------------------------------
-- Main

annotateImg :: Img -> String
annotateImg img = id
  $ svg img
  $ map (symRepr' img)
  $ symDetectAll img

decodeImg :: Img -> String
decodeImg img = id
    $ unlines
    $ map (intercalate "   ") -- join groups
    $ map (map (intercalate " ")) -- join items inside each group
    $ map (map (map (\(_, _, text, _) -> text)))
    $ map (groupBy (\a b -> xRight a >= xLeft b - 2)) -- split by horisontal groups
    $ splitByLines
    $ map (symRepr' img)
    $ symDetectAll img
  where
```

(continues on next page)

(continued from previous page)

```
    xLeft ((x, _), _, _, _) = x
    xRight ((x, _), (w, _), _, _) = x + w

splitByLines :: [(Coord, Size, a, b)] -> [[(Coord, Size, a, b)]]
splitByLines = id
  . map (sortOn (\((x, _), _, _, _) -> x))
  . map concat
  . map (map snd . snd) -- drop accumulators from both groupAcc's
  . groupAcc fst (\s x -> addRanges s (fst x))
  . groupAcc yRange (\s x -> addRanges s (yRange x))
  where
    yRange ((_, y), (_, h), _, _) = (y, y+h)
    addRanges (a0, a1) (b0, b1)
      | b0 <= a0 && a0 <= b1 = Just (b0, max a1 b1)
      | a0 <= b0 && b0 <= a1 = Just (a0, max a1 b1)
      | otherwise = Nothing

symRepr :: Symbol -> (String, String)
symRepr SymUnknown = ("?", "gray")
symRepr (SymSpecial x) = (x, "gray")
symRepr SymEllipsis = ("...", "gray")
symRepr (SymNumber val) = (show val, "green")
symRepr (SymModulatedNumber val) = ("[" ++ show val ++ "]", "purple")
symRepr (SymOperator val) = (text, "yellow")
  where
    text = fromMaybe (':' : show val) $ lookup val ops
    ops = [ (0, "ap")
          , (12, "=")
          -- constants
          , (2, "t")
          , (8, "f")
          -- unary operators
          , (401, "dec")
          , (417, "inc")
          , (170, "mod")
          , (341, "dem")
          -- binary operators
          , (40, "div")
          , (146, "mul")
          , (365, "add")
          -- comparisons
          , (416, "lt")
          , (448, "eq")
          ]
symRepr (SymBox w h val) = ('#' : show w ++ ":" ++ show h ++ ":" ++ showHex val "",
↪"orange")
symRepr (SymVariable val) = ('x' : show val, "blue")

symRepr' :: Img -> (Coord, Size) -> (Coord, Size, String, String)
symRepr' img (coord, size) =
  (coord, size, text, color)
  where (text, color) = symRepr $ symDecode img coord size

main :: IO ()
main = do
  [fnameIn, fnameSvg, fnameTxt] <- getArgs
  img <- imgLoad fnameIn 4
```

(continues on next page)

(continued from previous page)

```
  writeFile fnameSvg $ annotateImg img
  writeFile fnameTxt $ decodeImg img
```

CHAPTER

SIX

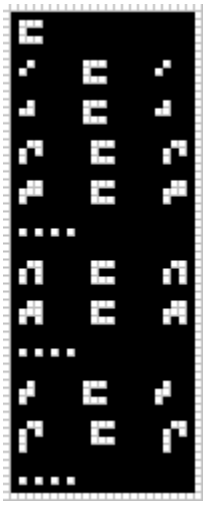
#4. EQUALITY

Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

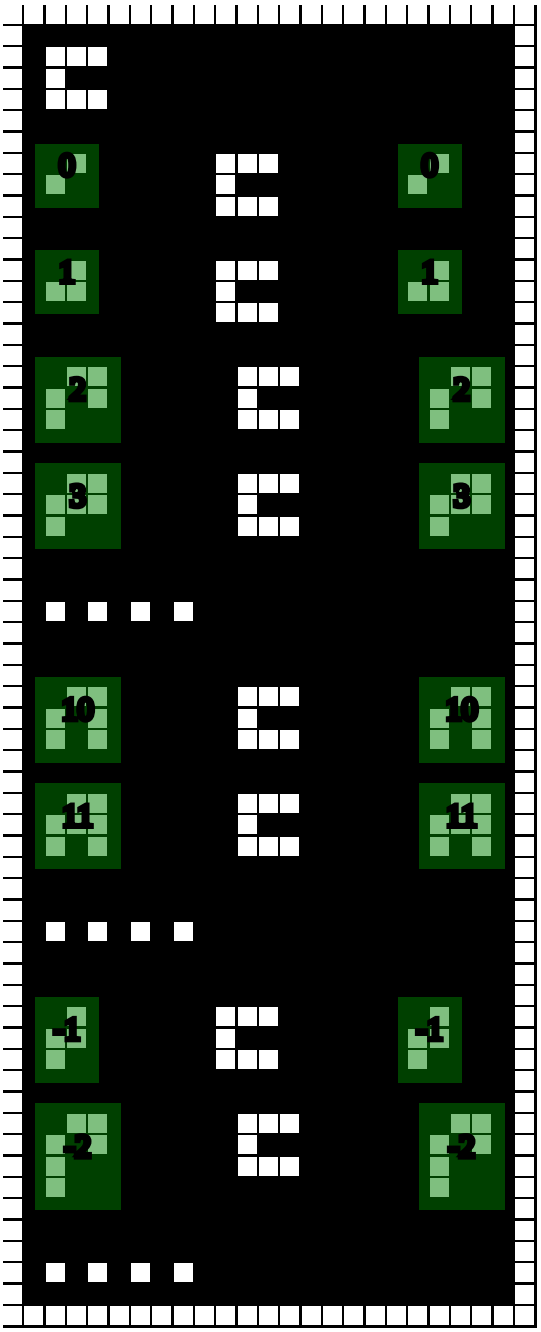
Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

6.1 Image

This image was produced from the fourth radio transmission using *previously contributed code*.



This partly annotated version of the image was made using *code from message #2*.


0
0
1
1
2
2
3
3
10
10
11
11
-1
-1
-2
-2

6.2 Interpretation

The new glyph is probably an equality sign, but there is not enough information be sure. Can be a less-than sign, any operation that preserves its operand, etc.

6.3 Decoded

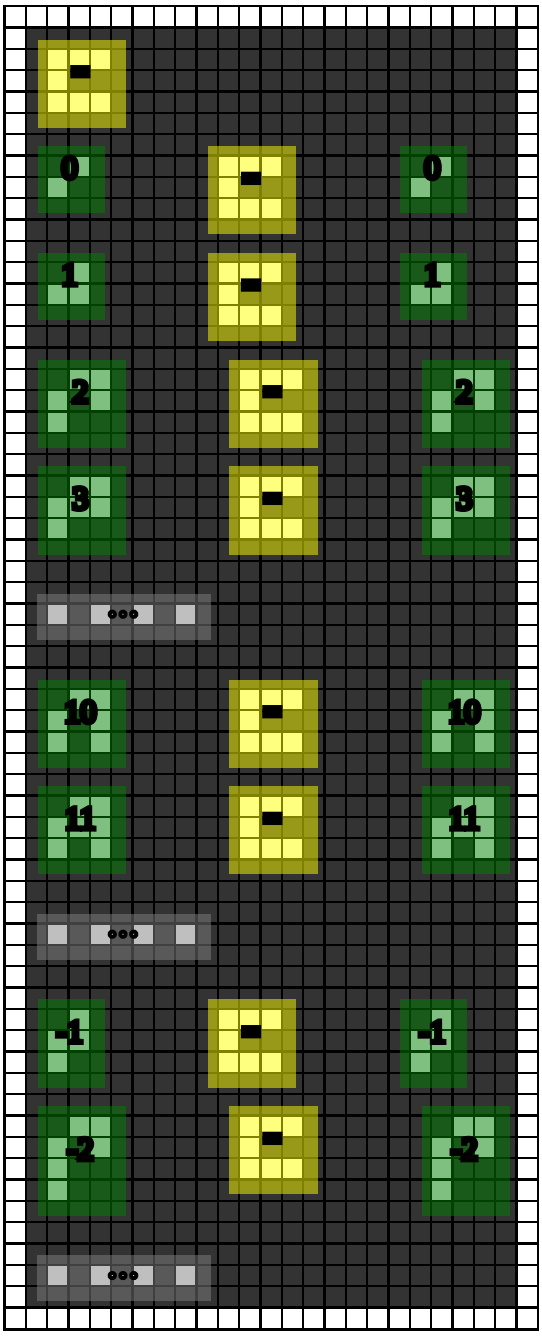
```
=
0   =   0
1   =   1
2   =   2
3   =   3
...
10   =   10
11   =   11
...
-1   =   -1
-2   =   -2
...
```

6.4 Code

Revised version of the Haskell code that supports the = glyph is published on the *message #3 page*.

Contributed by Discord user @pink_snow.

Example output:


0
0
1
1
2
2
3
3
10
10
11
11
-1
-1
-2
-2

CHAPTER

SEVEN

#5. SUCCESSOR

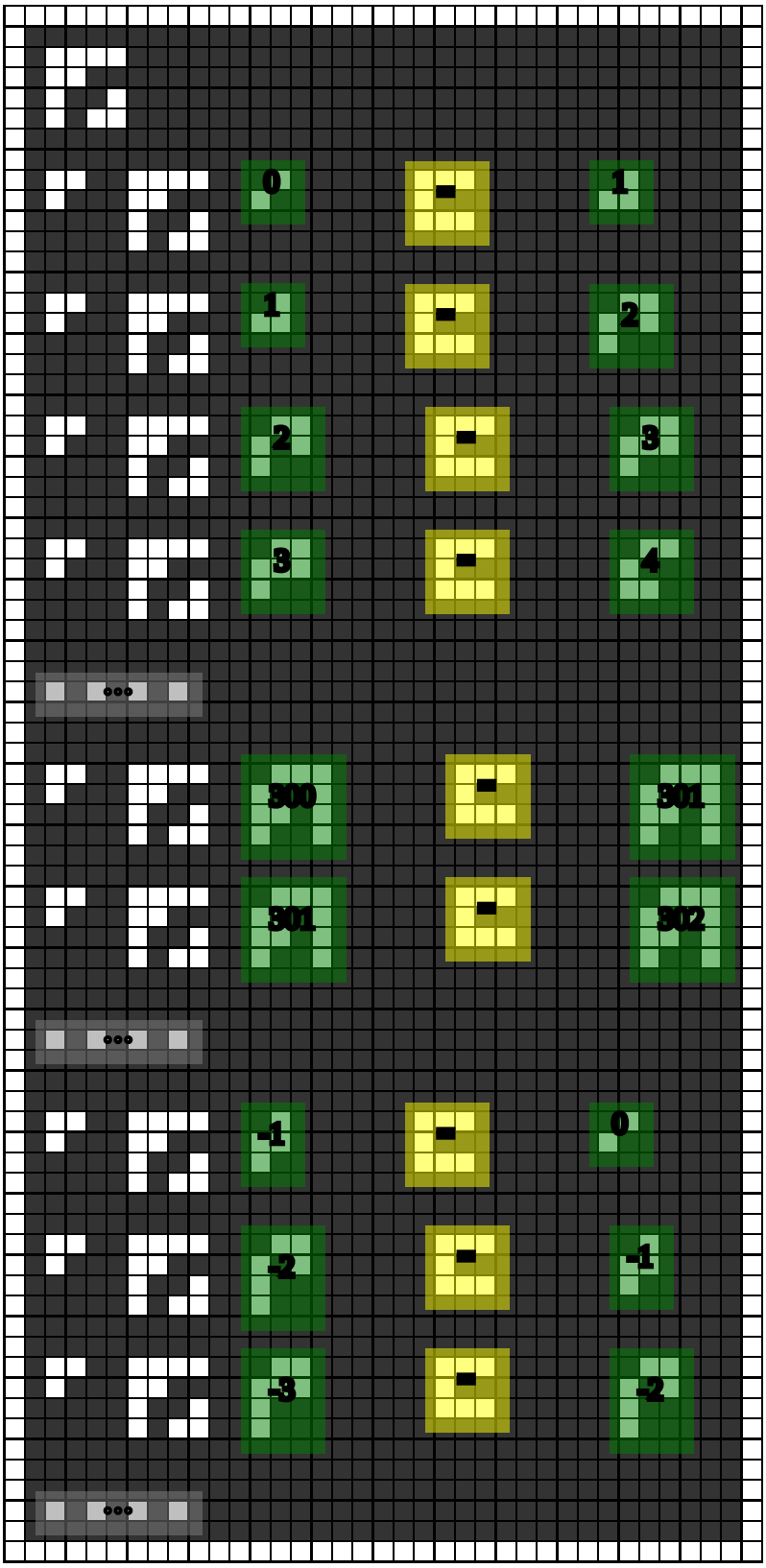
Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

7.1 Image

This image was produced from the fifth radio transmission using *previously contributed code*.

This partly annotated version of the image was made using *code from message #3*.

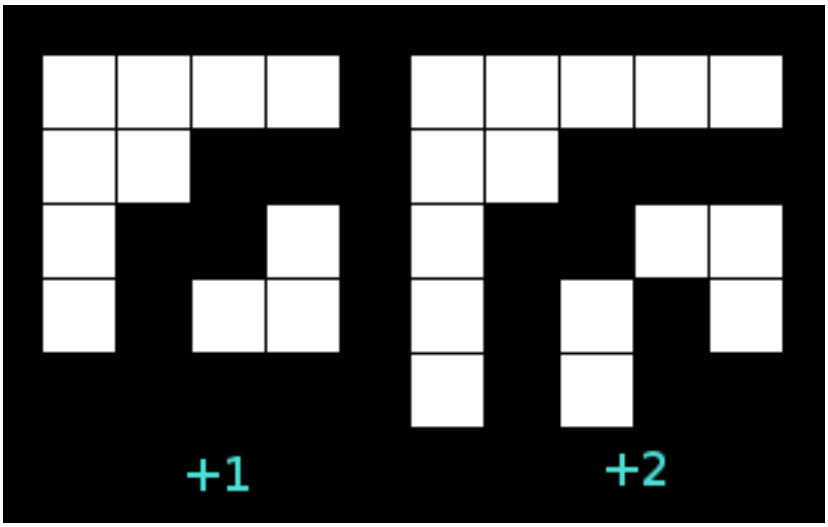

0
-
1
1
-
2
2
-
3
3
-
4
...
300
-
301
301
-
302
...
-1
-
0
-2
-
-1
-3
-
-2
...

7.2 Interpretation

There are two new symbols in this message, which are used inseparably from each other. It seems that a combination of them represents an increment operation.

The three-pixel symbol could be the application operator, and the other complicated one is the successor function. (by @nore)

The inner part of the complicated symbol is number 1. (by @gltronred) As a consequence of this observation, this symbol could contain any number: (image by @elventian)



The numerical value of new symbols are 0 and 417.

7.3 Decoded

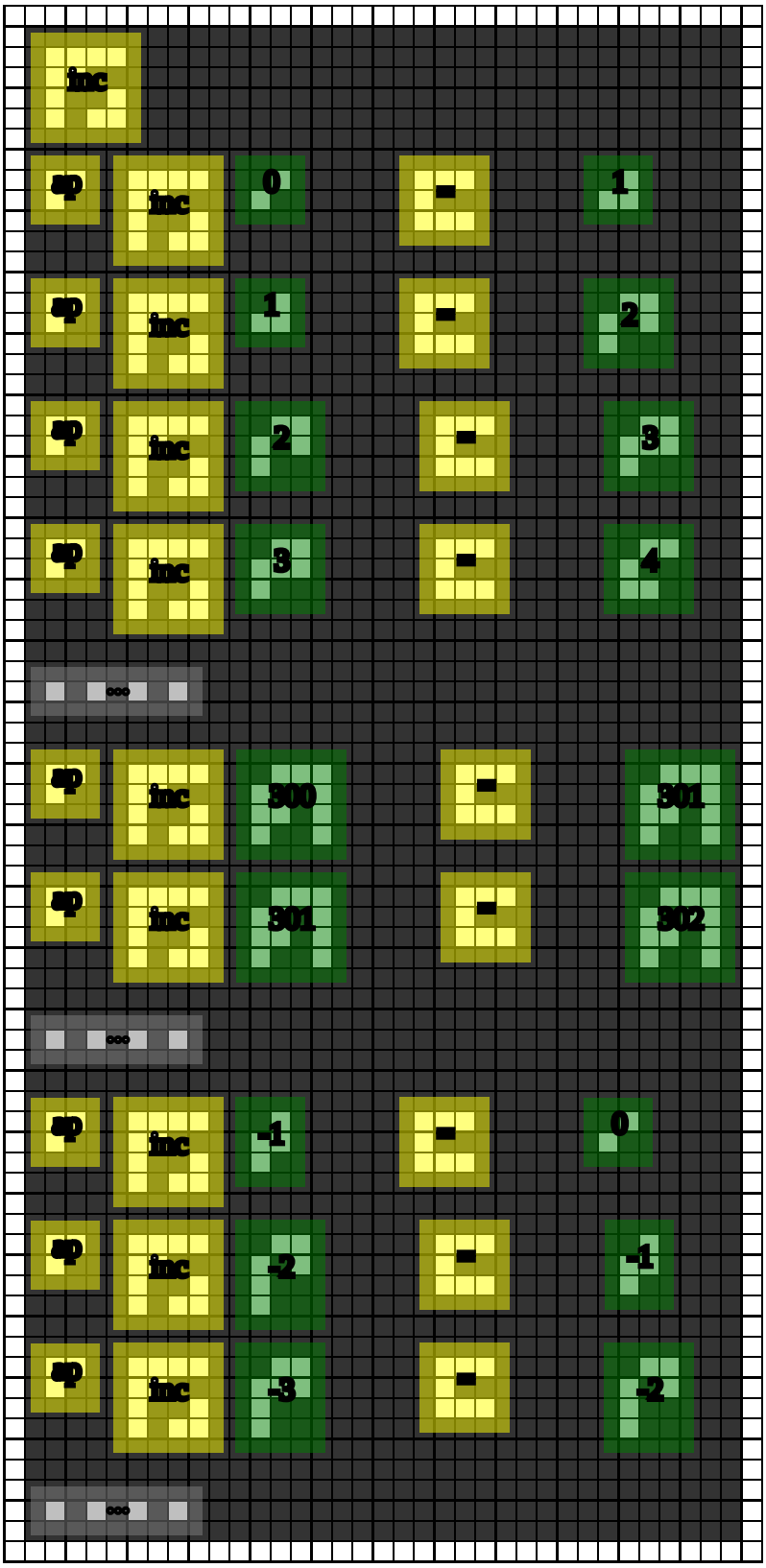
```
inc
ap inc 0    =    1
ap inc 1    =    2
ap inc 2    =    3
ap inc 3    =    4
...
ap inc 300    =    301
ap inc 301    =    302
...
ap inc -1    =    0
ap inc -2    =    -1
ap inc -3    =    -2
...
```

7.4 Code

Revised version of the Haskell code that supports the `ap` and `inc` glyphs is published on the *message #3 page*.

Contributed by Discord users @pink_snow and @fryguybob.

Example output:


inc
dp
inc
0
-
1
dp
inc
1
-
2
dp
inc
2
-
3
dp
inc
3
-
4
dp
inc
300
-
301
dp
inc
301
-
302
dp
inc
-1
-
0
dp
inc
-2
-
-1
dp
inc
-3
-
-2

CHAPTER

EIGHT

#6. PREDECESSOR

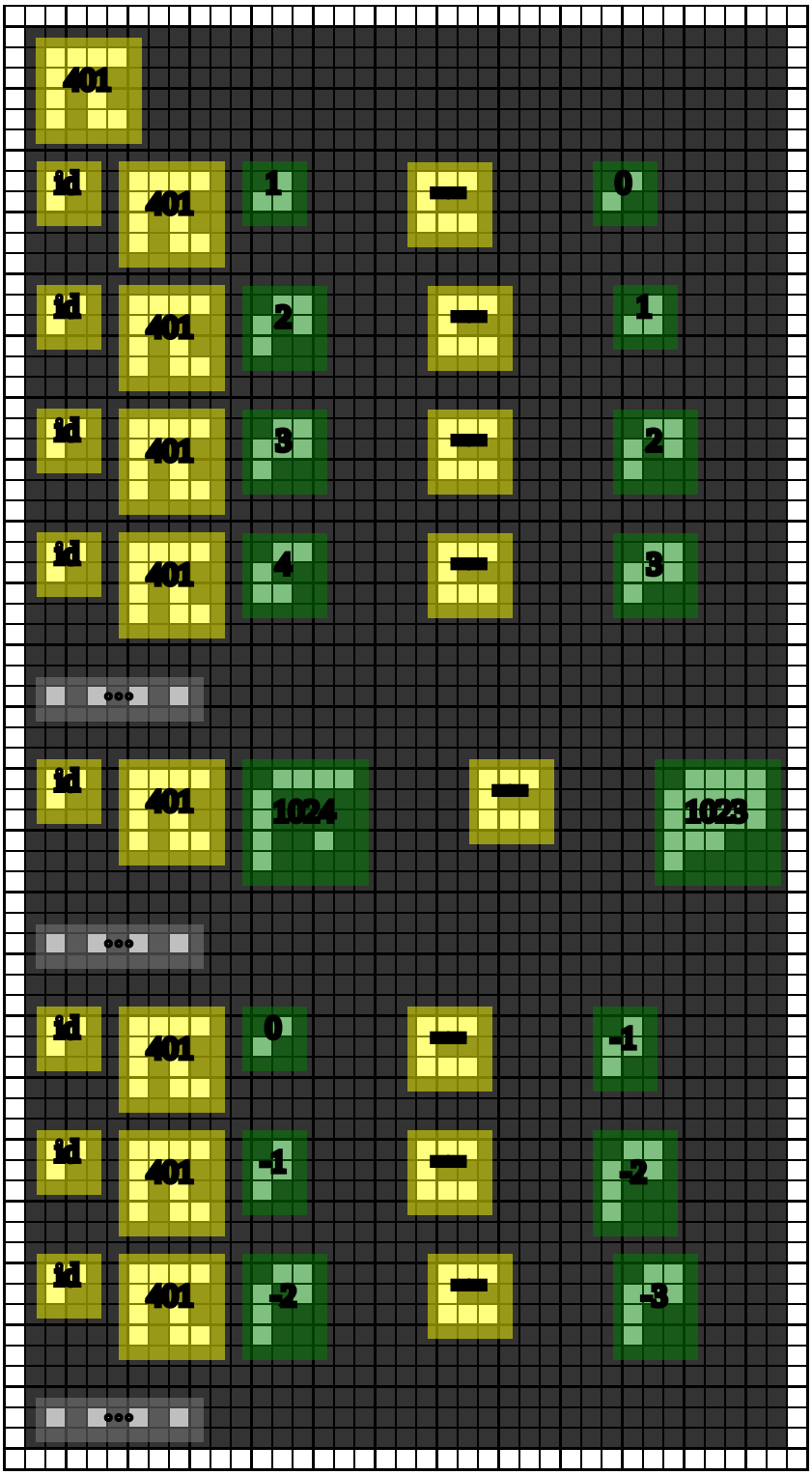
Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

8.1 Image

This image was produced from the sixth radio transmission using *previously contributed code*.

This partly annotated version of the image was made using *code from message #3*.


401
ld
401
1
—
0
ld
401
2
—
1
ld
401
3
—
2
ld
401
4
—
3
•••
ld
401
1024
—
1023
•••
ld
401
0
—
-1
ld
401
-1
—
-2
ld
401
-2
—
-3
•••

8.2 Interpretation

Contributed by Discord user @elventian.

Just like in *#5. Successor*, there are two unknown symbols in this message, together they represent a decrement operation.

The three-pixel symbol is identical to increment operation, and the other complicated one looks similar to increment, but rejects our theory about internal structure of the symbol, so it's just an arbitrary patterns.

8.3 Decoded

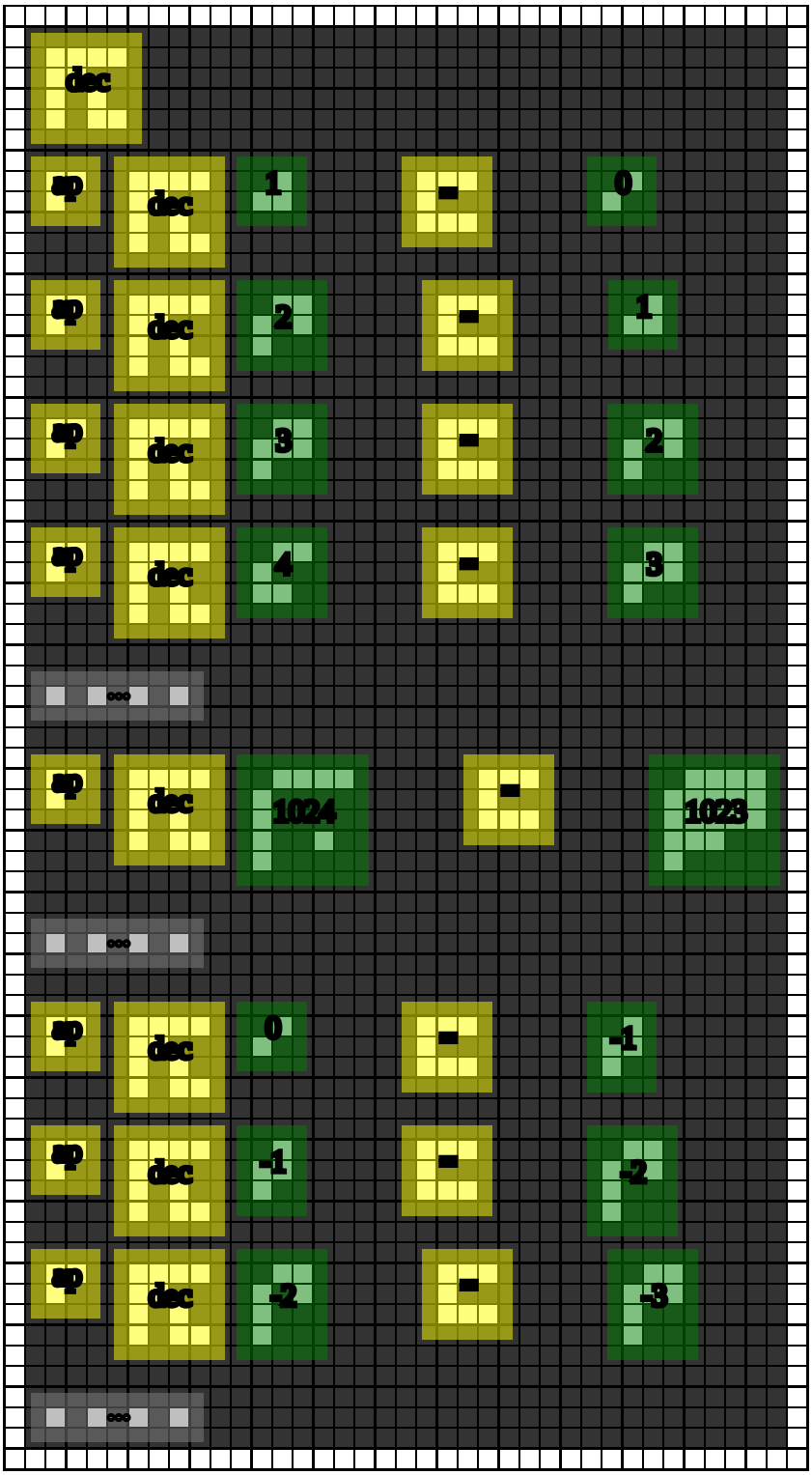
```
dec
ap dec 1    =    0
ap dec 2    =    1
ap dec 3    =    2
ap dec 4    =    3
...
ap dec 1024    =    1023
...
ap dec 0    =    -1
ap dec -1    =    -2
ap dec -2    =    -3
...
```

8.4 Code

Revised version of the Haskell code that supports the `dec` glyph is published on the *message #3 page*.

Contributed by Discord users @pink_snow and @fryguybob.

Example output:


dec
ap
dec
1
=
0
ap
dec
2
=
1
ap
dec
3
=
2
ap
dec
4
=
3
...
ap
dec
1024
=
1023
...
ap
dec
0
=
-1
ap
dec
-1
=
-2
ap
dec
-2
=
-3
...

CHAPTER

NINE

#7. SUM

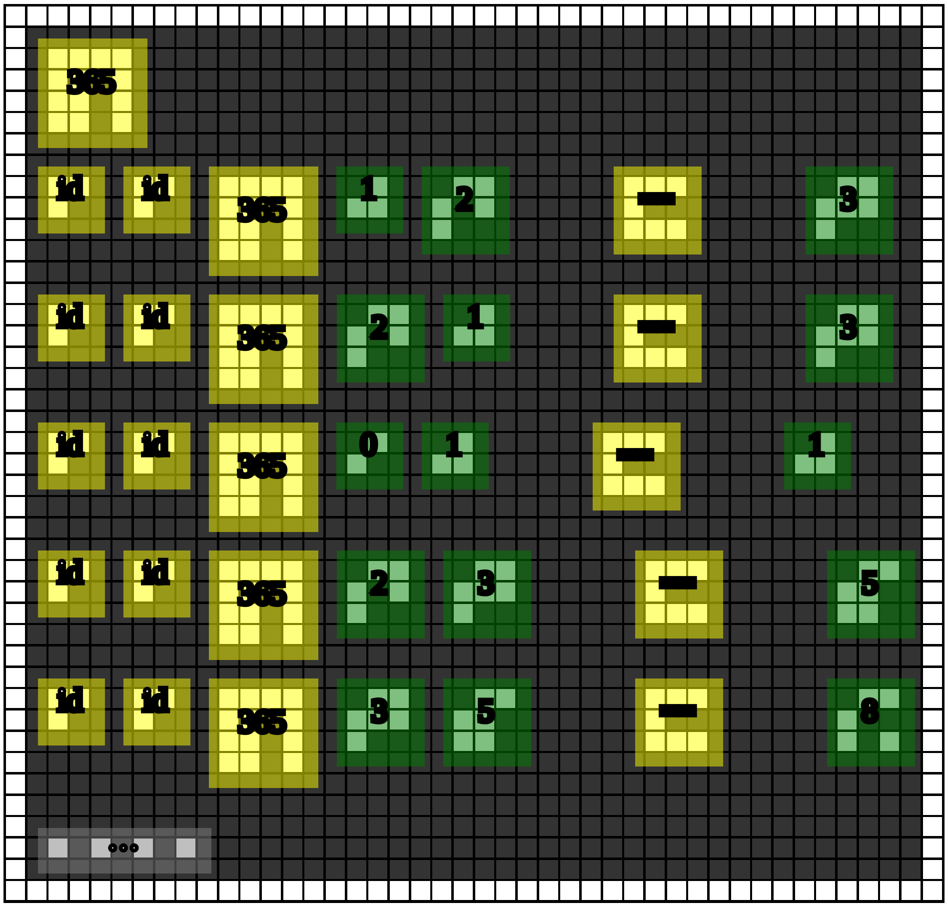
Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

9.1 Image

This image was produced from the seventh radio transmission using *previously contributed code*.

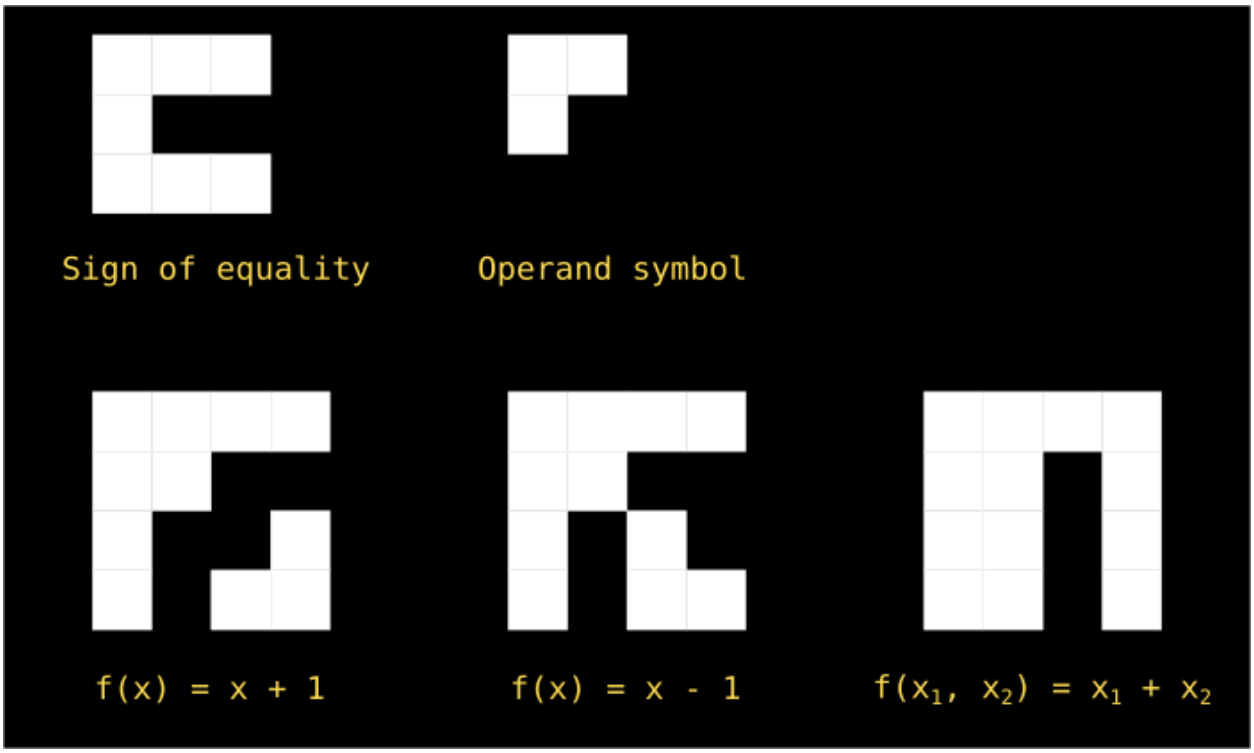
This partly annotated version of the image was made using *code from message #3*.



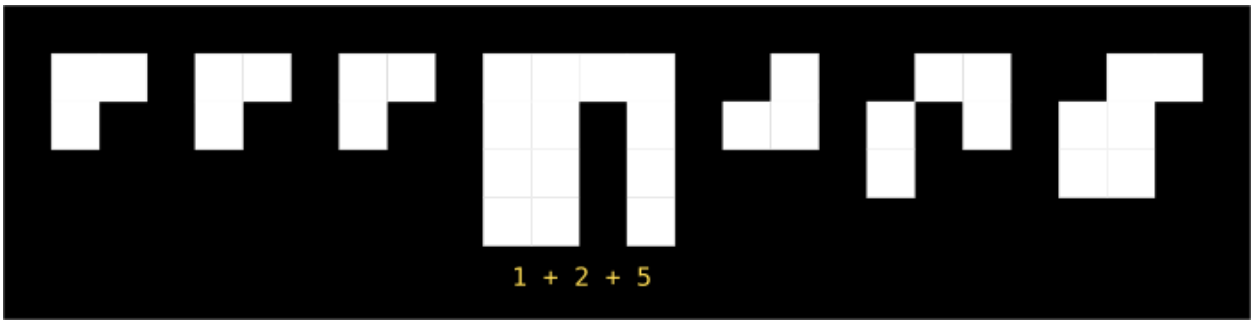
9.2 Interpretation

Contributed by Discord user @elventian.

This image shows all known operators and functions:



Count of operand symbols before function symbol defines how many operands the function expects. if this is correct, we need to do the following to calculate sum of three numbers:



9.3 Decoded

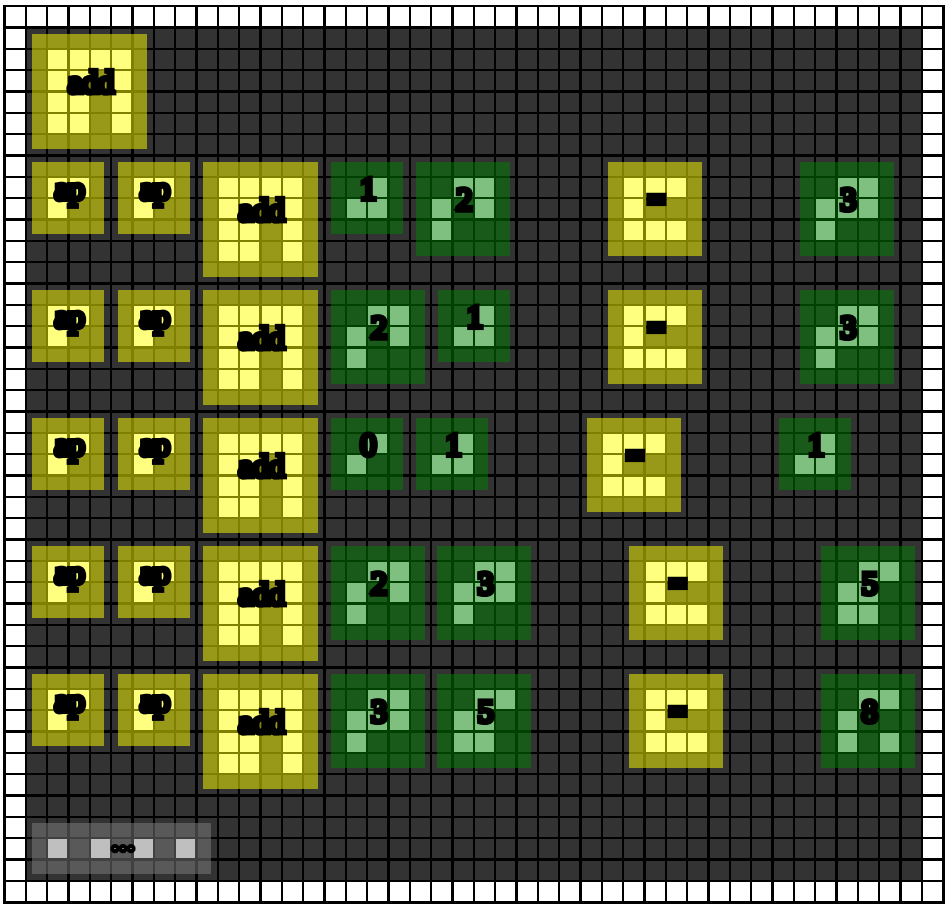
```
add
ap ap add 1 2    =    3
ap ap add 2 1    =    3
ap ap add 0 1    =    1
ap ap add 2 3    =    5
ap ap add 3 5    =    8
...
```

9.4 Code

Revised version of the Haskell code that supports the `add` glyph is published on the *message #3 page*.

Contributed by Discord users @pink_snow and @fryguybob.

Example output:



CHAPTER

TEN

#8. VARIABLES

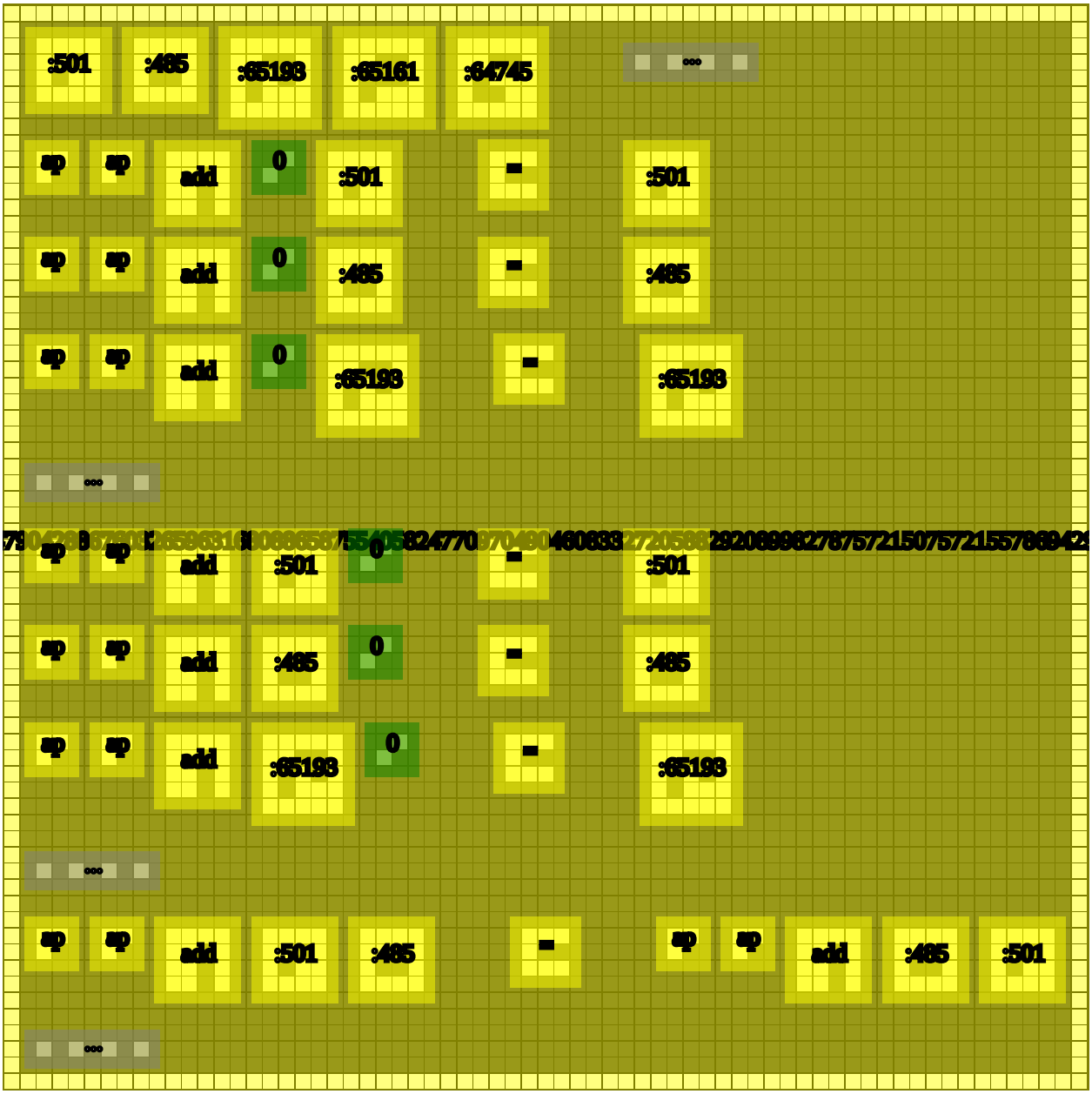
Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

10.1 Image

This image was produced from the eighth radio transmission using *previously contributed code*.

This partly annotated version of the image was made using *code from message #3*.



10.2 Interpretation

Contributed by Discord user @FiddlesticksMcGee.

Appears to designate a syntax for declaring variables.

Variables have a border of high values framing a negative of numbers as previously encoded in *#1. Numbers*.

10.3 Decoded

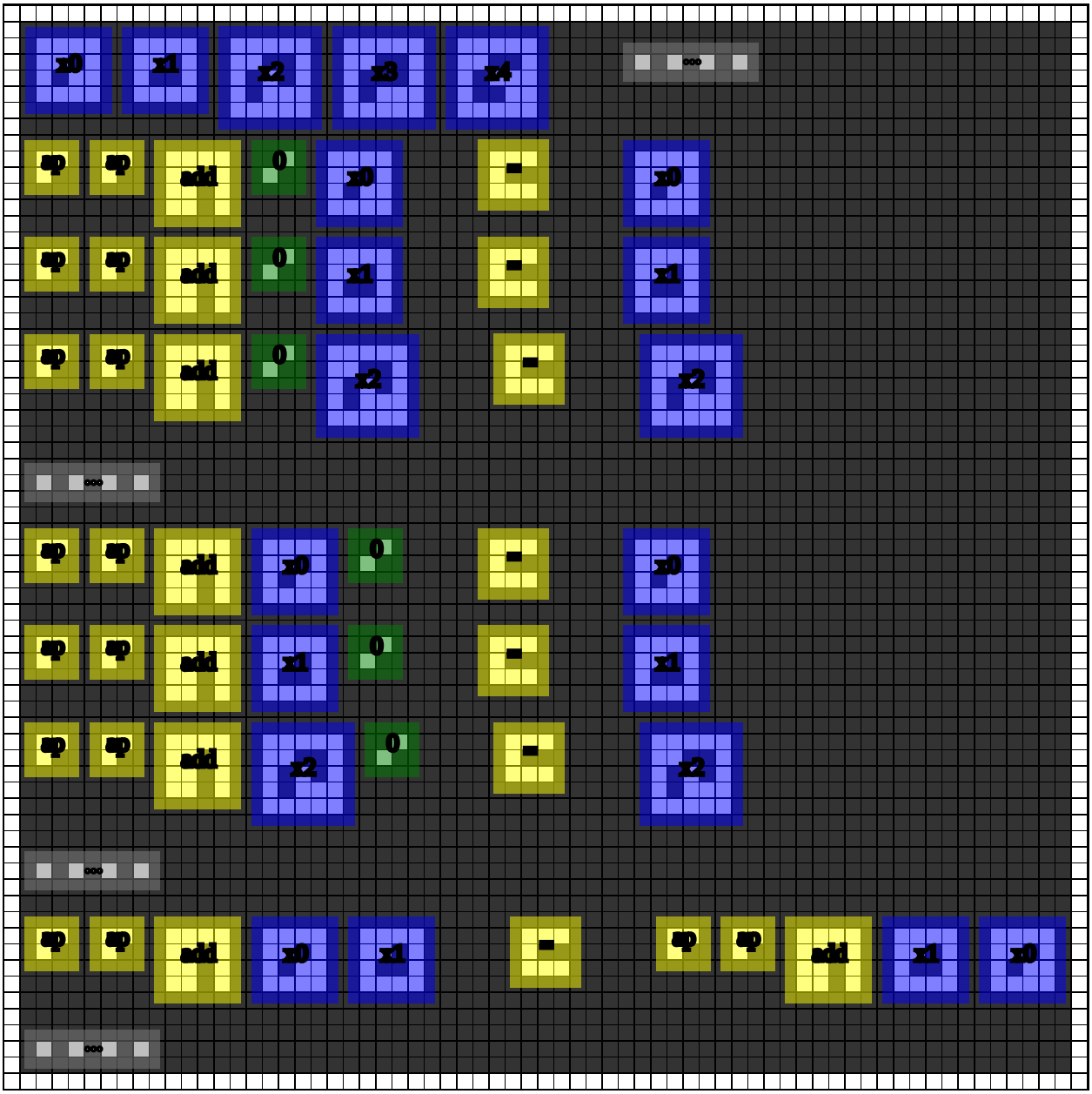
```
x0 x1 x2 x3 x4   ...
ap ap add 0 x0   =   x0
ap ap add 0 x1   =   x1
ap ap add 0 x2   =   x2
...
ap ap add x0 0   =   x0
ap ap add x1 0   =   x1
ap ap add x2 0   =   x2
...
ap ap add x0 x1   =   ap ap add x1 x0
...
```

10.4 Code

Revised version of the Haskell code that supports the variable glyphs is published on the *message #3 page*.

Contributed by Discord users @pink_snow and @fryguybob.

Example output:


x0
x1
x2
x3
x4
...
sp
sp
add
0
x0
-
x0
sp
sp
add
0
x1
-
x1
sp
sp
add
0
x2
-
x2
...
sp
sp
add
x0
0
-
x0
sp
sp
add
x1
0
-
x1
sp
sp
add
x2
0
-
x2
...
sp
sp
add
x0
x1
-
sp
sp
add
x1
x0
...

CHAPTER

ELEVEN

#9. PRODUCT

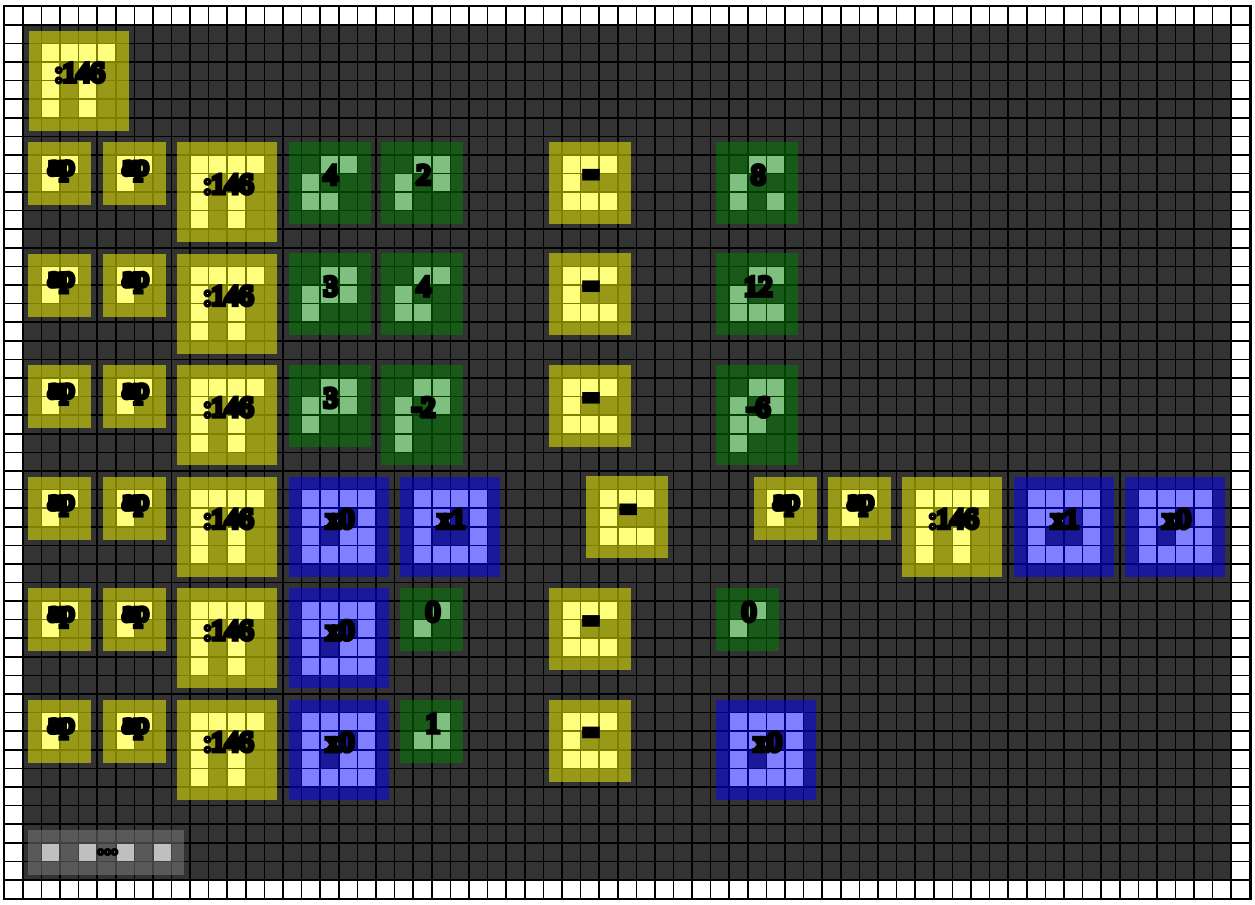
Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

11.1 Image

This image was produced from the ninth radio transmission using *previously contributed code*.

This partly annotated version of the image was made using *code from message #3*.



11.2 Interpretation

Definition of the new glyph is consistent with the definition of multiplication.

11.3 Decoded

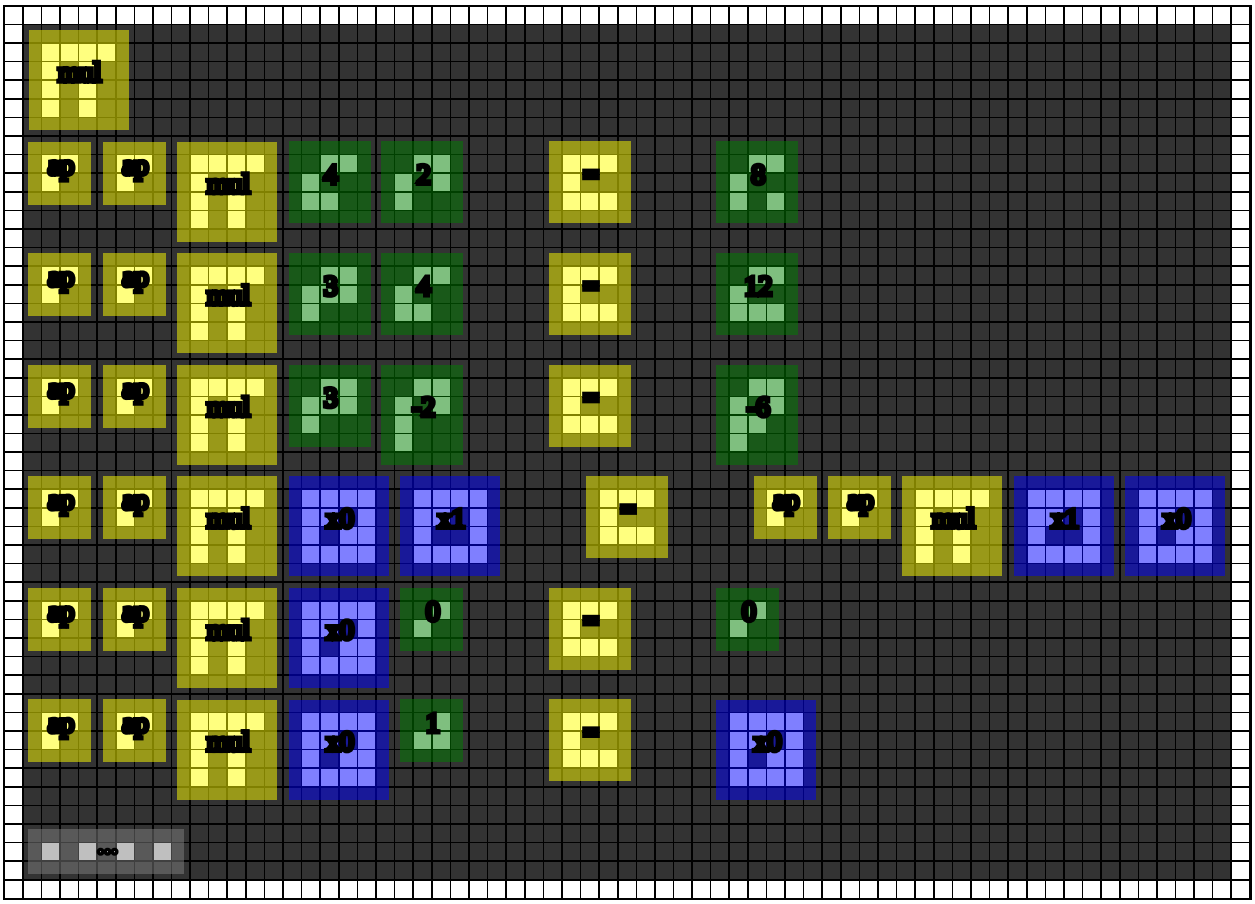
```
mul
ap ap mul 4 2    =    8
ap ap mul 3 4    =    12
ap ap mul 3 -2    =    -6
ap ap mul x0 x1    =    ap ap mul x1 x0
ap ap mul x0 0    =    0
ap ap mul x0 1    =    x0
...
```

11.4 Code

Revised version of the Haskell code that supports the `mul` glyph is published on the *message #3 page*.

Contributed by Discord users @pink_snow and @fryguybob.

Example output:



CHAPTER

TWELVE

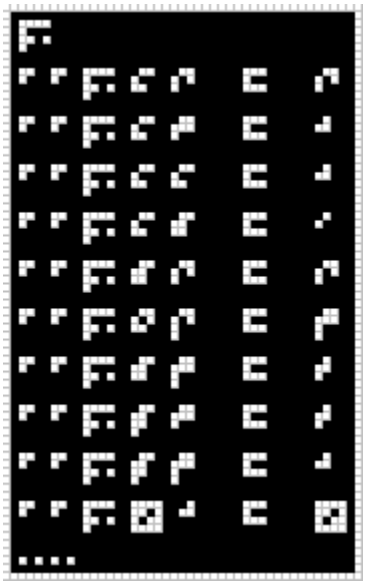
#10. INTEGER DIVISION

Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

12.1 Image

This image was produced from the tenth radio transmission using *previously contributed code*.



This partly annotated version of the image was made using *code from message #3*.


:40
4
2
2
4
3
1
4
4
1
4
5
0
5
2
2
6
-2
-3
5
-3
-1
-5
3
-1
-5
-3
1
1
:0
:0
:00

12.2 Interpretation

The new operator is consistent with integer division which rounds toward zero.

12.3 Decoded

```
div
ap ap div 4 2    =    2
ap ap div 4 3    =    1
ap ap div 4 4    =    1
ap ap div 4 5    =    0
ap ap div 5 2    =    2
ap ap div 6 -2    =    -3
ap ap div 5 -3    =    -1
ap ap div -5 3    =    -1
ap ap div -5 -3    =    1
ap ap div x0 1    =    x0
...
```

12.4 Code

The *Haskell code* has been revised to decode the integer division operator.

```
diff --git a/source/annotate.hs b/source/annotate.hs
index 6a1e50e..21c4c11 100755
--- a/source/annotate.hs
+++ b/source/annotate.hs
@@ -273,6 +273,7 @@ symRepr (SymNumber val) = (show val, "green")
 symRepr (SymOperator val) = (t val, "yellow")
   where t 0 = "ap"
         t 12 = "="
+        t 40 = "div"
         t 146 = "mul"
         t 401 = "dec"
         t 417 = "inc"
```

Example output:


div
sp sp
div
4 2
-
2
sp sp
div
4 3
-
1
sp sp
div
4 4
-
1
sp sp
div
4 5
-
0
sp sp
div
5 2
-
2
sp sp
div
6 -2
-
-3
sp sp
div
5 -3
-
-1
sp sp
div
-5 3
-
-1
sp sp
div
-5 -3
-
1
sp sp
div
x0 1
-
x0

CHAPTER

THIRTEEN

#11. EQUALITY AND BOOLEANS

Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

13.1 Image

This image was produced from the eleventh radio transmission using *previously contributed code*.

This partly annotated version of the image was made using *code from message #3*.

13.2 Interpretation

Operator 448 is consistent with checking whether the first number equal than the second number. We will denote it as *eq*.

Operator 2 is consistent with true and operator 8 is consistent with false. We will denote it as *t* and *f* respectively.

13.3 Decoded

```
eq
ap ap eq x0 x0   =   t
ap ap eq 0 -2    =   f
ap ap eq 0 -1    =   f
ap ap eq 0 0     =   t
ap ap eq 0 1     =   f
ap ap eq 0 2     =   f
...
ap ap eq 1 -1    =   f
ap ap eq 1 0     =   f
ap ap eq 1 1     =   t
ap ap eq 1 2     =   f
ap ap eq 1 3     =   f
...
ap ap eq 2 0     =   f
ap ap eq 2 1     =   f
ap ap eq 2 2     =   t
ap ap eq 2 3     =   f
ap ap eq 2 4     =   f
...
ap ap eq 19 20   =   f
ap ap eq 20 20   =   t
ap ap eq 21 20   =   f
...
ap ap eq -19 -20   =   f
ap ap eq -20 -20   =   t
ap ap eq -21 -20   =   f
...
```

13.4 Code

The *Haskell code* has been revised to decode new glyphs.

Example output:

CHAPTER

FOURTEEN

#12. STRICT LESS-THAN

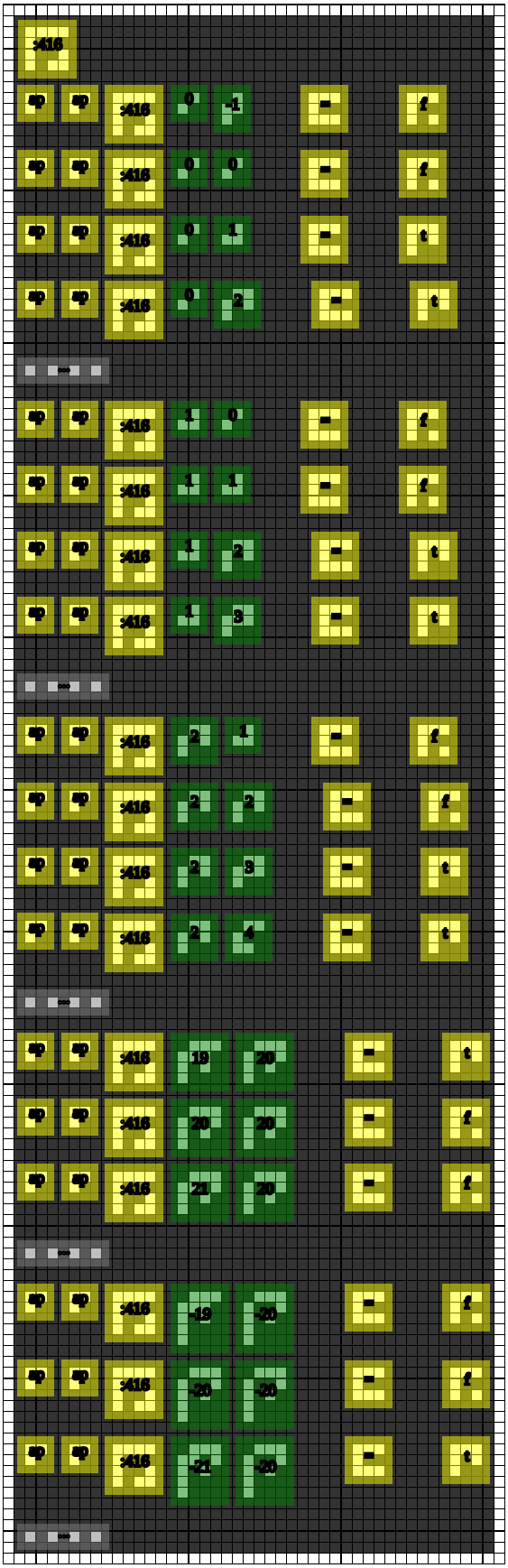
Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

14.1 Image

This image was produced from the twelfth radio transmission using *previously contributed code*.

This partly annotated version of the image was made using *code from message #3*.



14.2 Interpretation

Operator 416 is consistent with checking whether the first number is strictly less than the second number. We will denote it as *lt*.

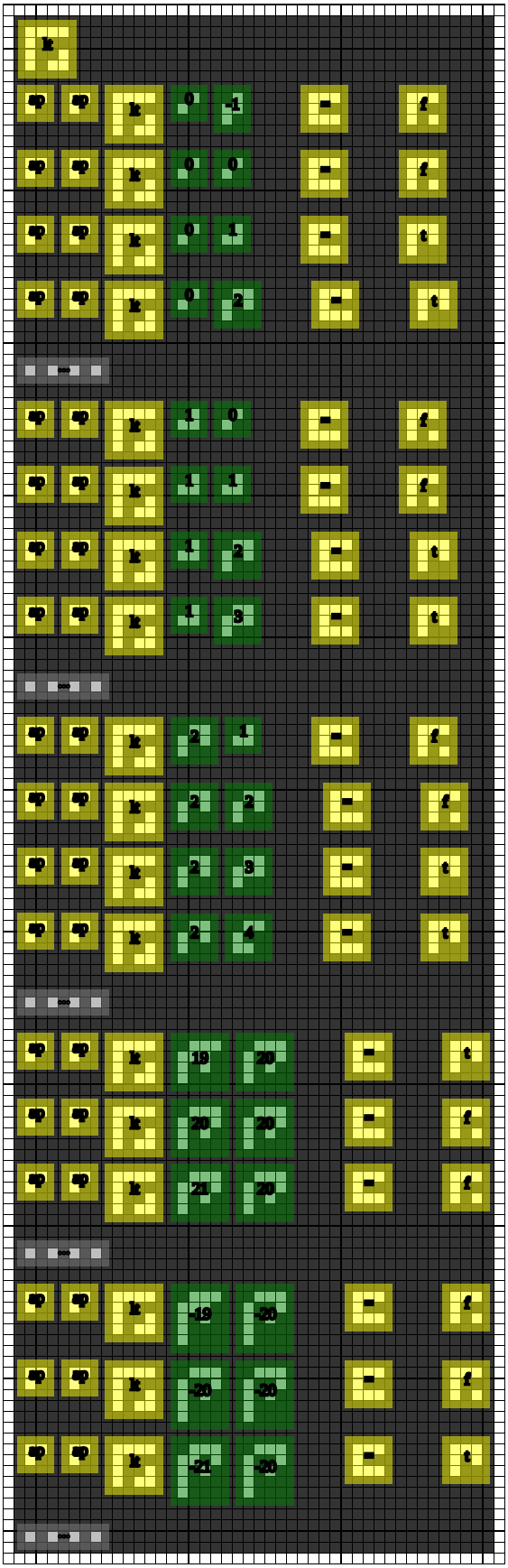
14.3 Decoded

```
lt
ap ap lt 0 -1   =   f
ap ap lt 0 0    =   f
ap ap lt 0 1    =   t
ap ap lt 0 2    =   t
...
ap ap lt 1 0    =   f
ap ap lt 1 1    =   f
ap ap lt 1 2    =   t
ap ap lt 1 3    =   t
...
ap ap lt 2 1    =   f
ap ap lt 2 2    =   f
ap ap lt 2 3    =   t
ap ap lt 2 4    =   t
...
ap ap lt 19 20   =   t
ap ap lt 20 20   =   f
ap ap lt 21 20   =   f
...
ap ap lt -19 -20   =   f
ap ap lt -20 -20   =   f
ap ap lt -21 -20   =   t
...
```

14.4 Code

The *Haskell code* has been revised to decode new glyphs.

Example output:



CHAPTER

FIFTEEN

#13. MODULATE

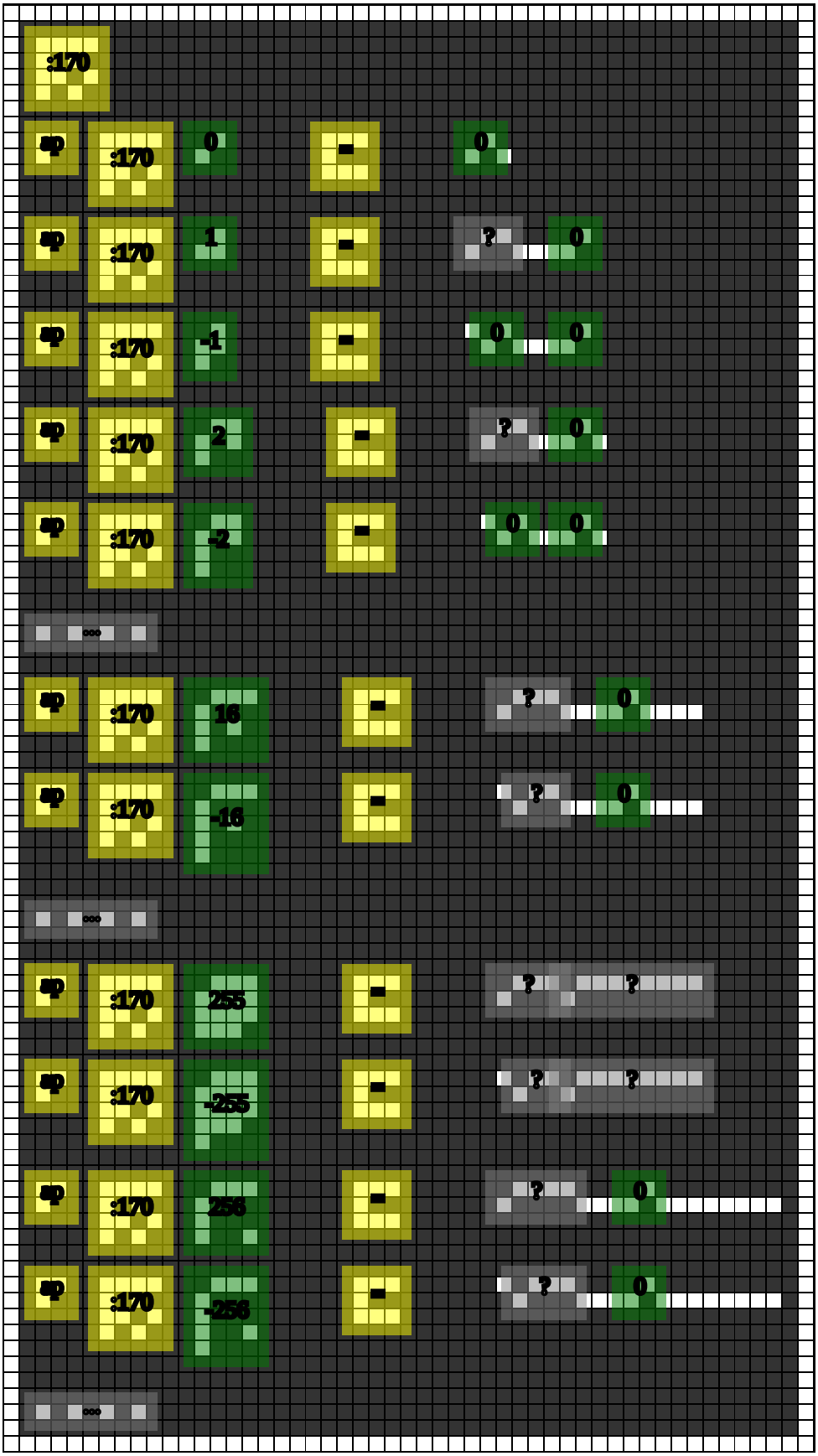
Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

15.1 Image

This image was produced from the thirteenth radio transmission using *previously contributed code*.

This partly annotated version of the image was made using *code from message #3*.


:170
ap
:170
0
=
0
ap
:170
1
=
?
0
ap
:170
-1
=
0
0
ap
:170
2
=
?
0
ap
:170
-2
=
0
0
...
ap
:170
16
=
?
0
ap
:170
-16
=
?
0
...
ap
:170
255
=
?
?
ap
:170
-255
=
?
?
ap
:170
256
=
?
0
ap
:170
-256
=
?
0
...

15.2 Interpretation

The operator defined in this message, `mod`, is for converting numbers from a grid form into a linear-encoded form. The linear encoding appears to be a type of Variable-length encoding, with the following form:

- Bits 0..1 define a positive or negative number (and signal width) via a high/low or low/high signal change:
  - 01: positive number
  - 10: negative number
- Bits 2..(n+2) define the width of the following binary-encoded number via a unary-encoded number of length *n* composed of high signals ending with a low signal. The number width (in bits) is four times the unary encoding (i.e. 4 * *n*):
  - 0: 0 [i.e. the number zero]
  - 10: 4-bit number [i.e. 1-15]
  - 110: 8-bit number [i.e. 1-255]
  - 1110: 12-bit number [i.e. 1-4095]
  - ...
- The remaining bits, i.e. (n + 3)..(n + 3 + 4*n - 1), determine the number itself, in most-significant-bit first binary notation. Using the examples from this message:
  - 0001: 1
  - 00010000: 16
  - 000100000000: 256
  - ...

With this encoding, the number zero only requires three bits (i.e. 010), but arbitrarily large numbers can also be represented.

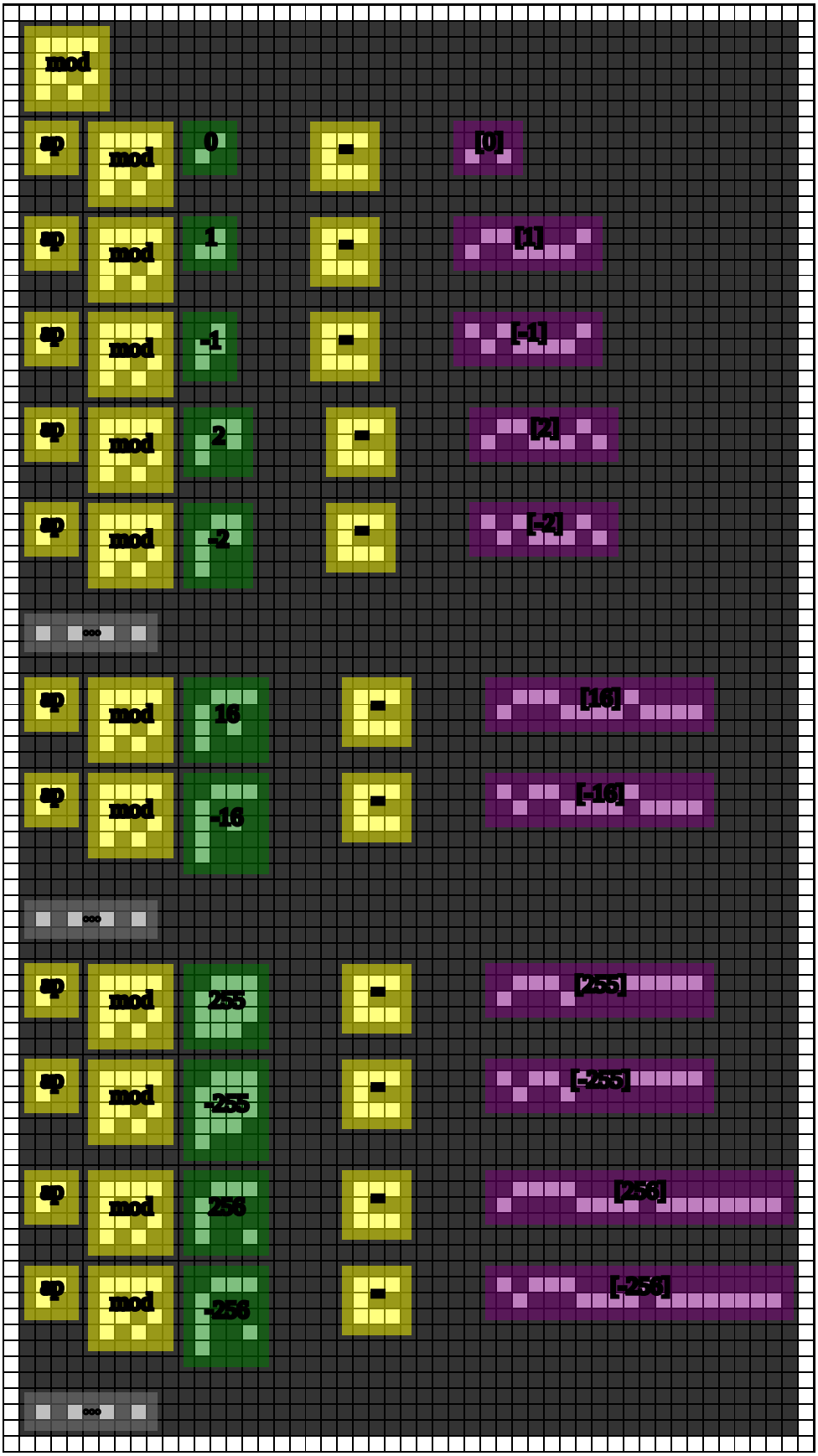
15.3 Decoded

```
mod
ap mod 0    =    [0]
ap mod 1    =    [1]
ap mod -1    =    [-1]
ap mod 2    =    [2]
ap mod -2    =    [-2]
...
ap mod 16    =    [16]
ap mod -16    =    [-16]
...
ap mod 255    =    [255]
ap mod -255    =    [-255]
ap mod 256    =    [256]
ap mod -256    =    [-256]
...
```

15.4 Code

The *Haskell code* has been revised to decode new glyphs.

Example output:


mod
sp
mod
0
=
[0]
sp
mod
1
=
[1]
sp
mod
-1
=
[-1]
sp
mod
2
=
[2]
sp
mod
-2
=
[-2]
...
sp
mod
16
=
[16]
sp
mod
-16
=
[-16]
...
sp
mod
255
=
[255]
sp
mod
-255
=
[-255]
sp
mod
256
=
[256]
sp
mod
-256
=
[-256]
...

CHAPTER

SIXTEEN

#14. DEMODULATE

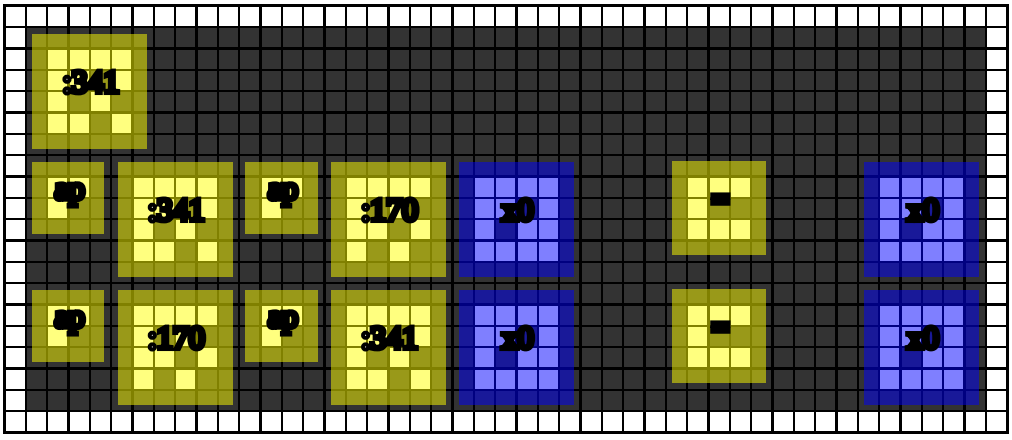
Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

16.1 Image

This image was produced from the fourteenth radio transmission using *previously contributed code*.

This partly annotated version of the image was made using *code from message #3*.



16.2 Interpretation

This appears to define an inverse operator to that from *#13. Modulate*, demonstrating that `:341` (`dem`) is the inverse of `:170` (`mod`), and vice versa. This suggests that `:341`, when applied to a linear-encoded number, will create a grid-encoded number.

The behaviour of `mod` when applied to a linear number, or `dem` when applied to a grid number, is *not* defined.

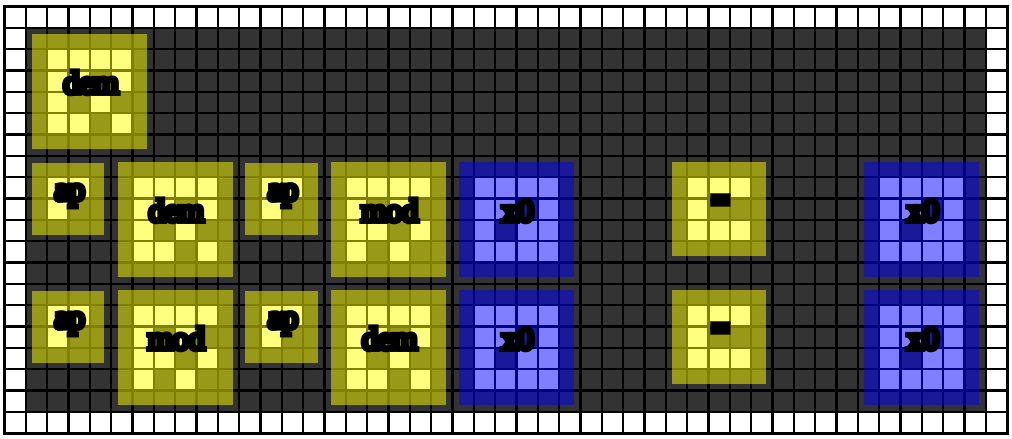
16.3 Decoded

```
dem
ap dem ap mod x0    =    x0
ap mod ap dem x0    =    x0
```

16.4 Code

The *Haskell code* has been revised to decode new glyphs.

Example output:



CHAPTER

SEVENTEEN

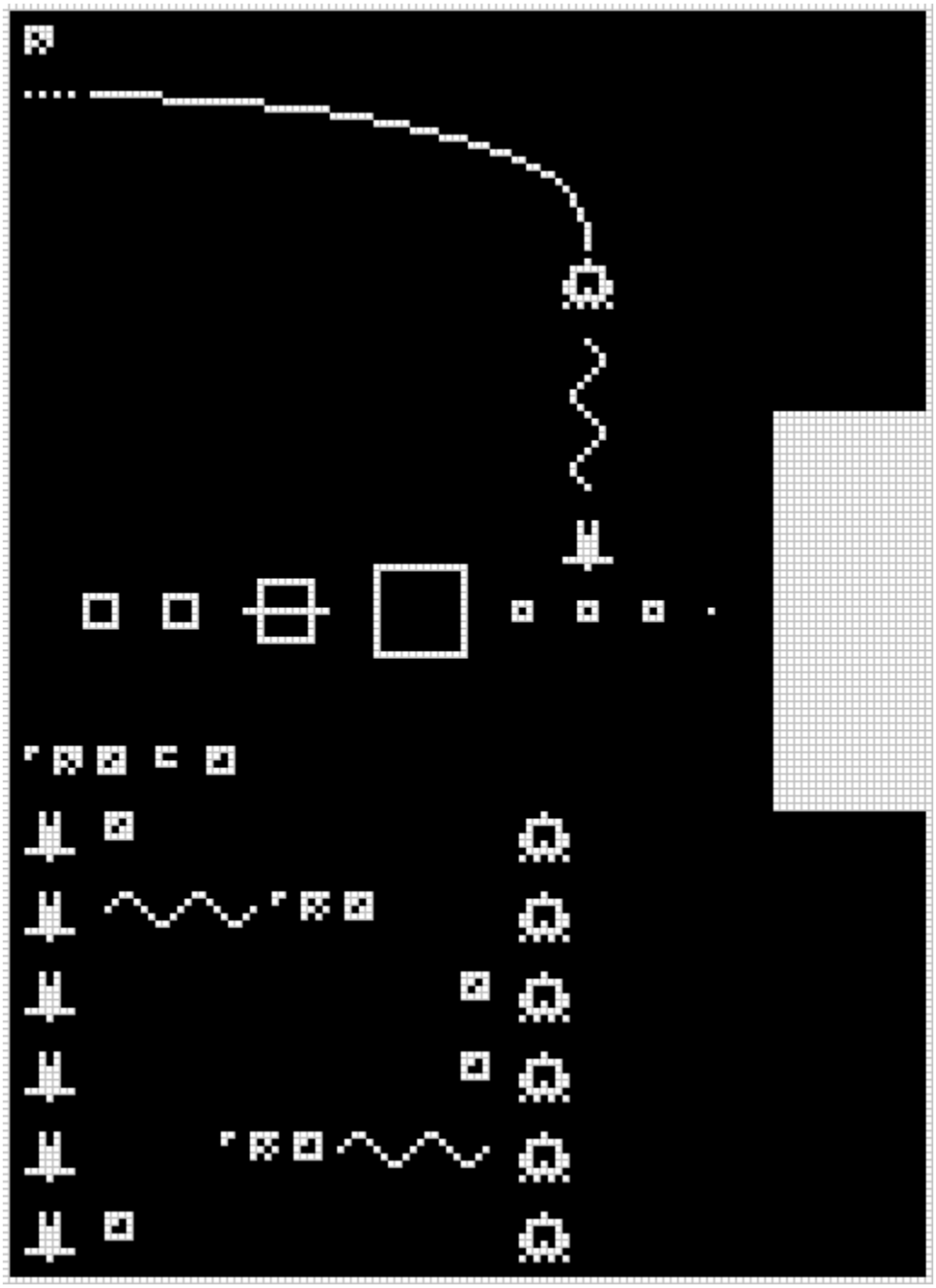
#15. SEND

Note: If you have any ideas or enhancements for this page, please edit it on GitHub!

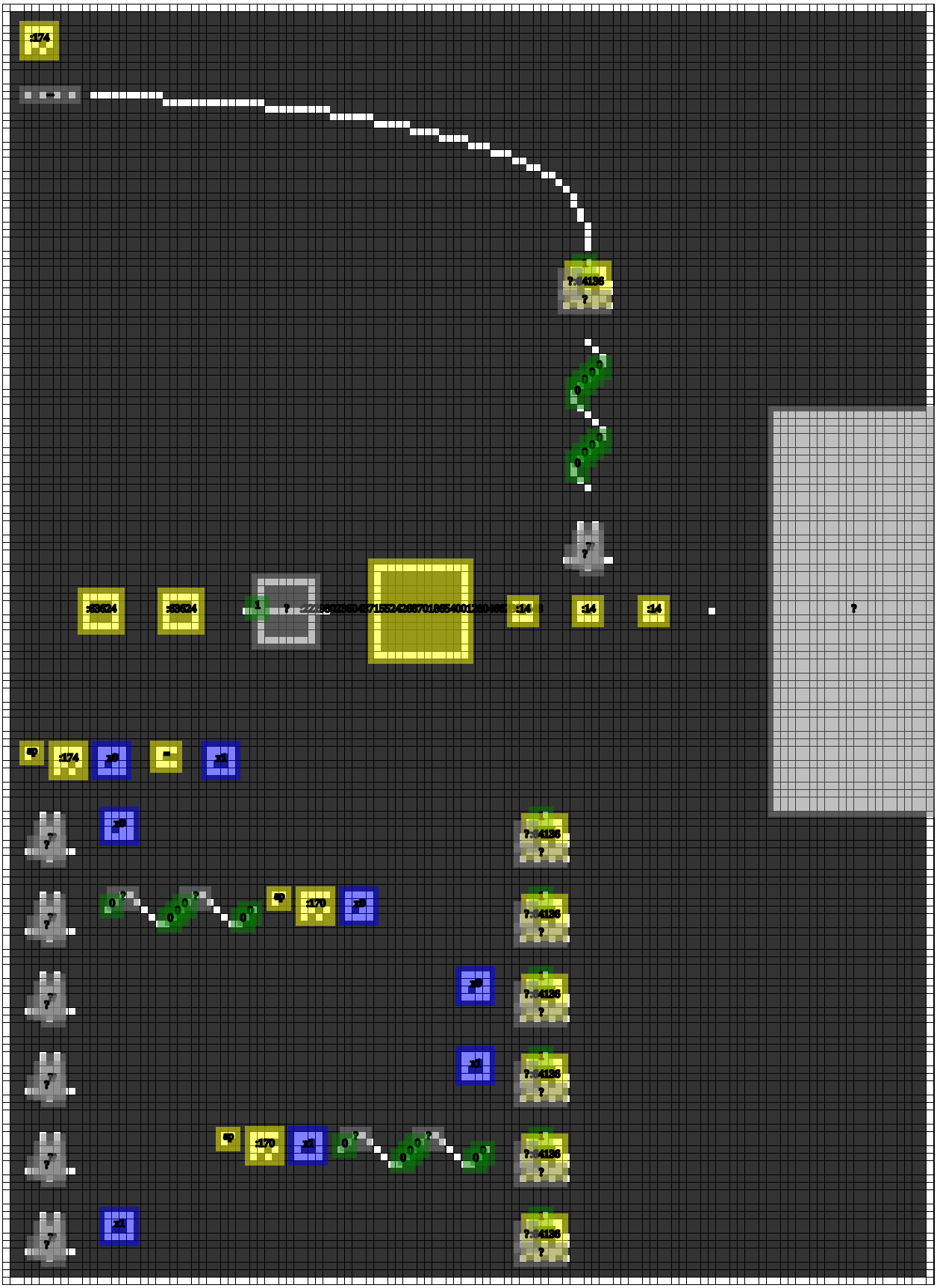
Following documentation is a cooperative result combined from our Discord chat and numerous pull requests. Thanks to everyone who helped!

17.1 Image

This image was produced from the fifteenth radio transmission using *previously contributed code*.



This partly annotated version of the image was made using *code from message #3*.



17.2 Interpretation

Todo: Add an interpretation for the fifteenth message.

17.3 Decoded

```
send

ap send x0 =  x1

humans   x0                      aliens
humans   ~~~~~ ap mod x0         aliens
humans                        x0 aliens
humans                        x1 aliens
humans          ap mod x1 ~~~~~ aliens
humans x1                        aliens
```

17.4 Code

Todo: Revise the *Haskell code* to support new glyphs from the fifteenth message.

CHAPTER

EIGHTEEN

#16. NEGATE

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

18.1 Image

This image was produced from the sixteenth radio transmission using *previously contributed code*.

18.2 Interpretation

18.3 Decoded

```
neg
ap neg 0    =    0
ap neg 1    =    -1
ap neg -1    =    1
ap neg 2    =    -2
ap neg -2    =    2
...
```

CHAPTER

NINETEEN

#17. FUNCTION APPLICATION

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

19.1 Image

This image was produced from the seventeenth radio transmission using *previously contributed code*.

19.2 Interpretation

`ap f x` is f(x)

The last two lines demonstrate that function application *ap* allows curried (i.e. partially evaluated) functions, by defining *inc* as the function *x -> 1 + x*.

19.3 Decoded

```
ap
ap inc ap inc 0   =   2
ap inc ap inc ap inc 0   =   3
ap inc ap dec x0   =   x0
ap dec ap inc x0   =   x0
ap dec ap ap add x0 1   =   x0
ap ap add ap ap add 2 3 4   =   9
ap ap add 2 ap ap add 3 4   =   9
ap ap add ap ap mul 2 3 4   =   10
ap ap mul 2 ap ap add 3 4   =   14
inc   =   ap add 1
dec   =   ap add ap neg 1
...
```

CHAPTER

TWENTY

#18. S COMBINATOR

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

20.1 Image

This image was produced from the eighteenth radio transmission using *previously contributed code*.

20.2 Interpretation

See https://en.wikipedia.org/wiki/B,_C,_K,_W_system

20.3 Decoded

```
s
ap ap ap s x0 x1 x2   =   ap ap x0 x2 ap x1 x2
ap ap ap s add inc 1   =   3
ap ap ap s mul ap add 1 6   =   42
...
```

CHAPTER

TWENTYONE

#19. C COMBINATOR

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

21.1 Image

This image was produced from the nineteenth radio transmission using *previously contributed code*.

21.2 Interpretation

See https://en.wikipedia.org/wiki/B,_C,_K,_W_system

21.3 Decoded

```
c
ap ap ap c x0 x1 x2   =   ap ap x0 x2 x1
ap ap ap c add 1 2   =   3
...
```

CHAPTER

TWENTYTWO

#20. B COMBINATOR

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

22.1 Image

This image was produced from the twentieth radio transmission using *previously contributed code*.

22.2 Interpretation

See https://en.wikipedia.org/wiki/B,_C,_K,_W_system

22.3 Decoded

```
b
ap ap ap b x0 x1 x2   =   ap x0 ap x1 x2
ap ap ap b inc dec x0   =   x0
...
```

CHAPTER

TWENTYTHREE

#21. TRUE (K COMBINATOR)

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

23.1 Image

This image was produced from the twenty-first radio transmission using *previously contributed code*.

23.2 Interpretation

Decoded as `t`, because it has a meaning of boolean True.

23.3 Decoded

```
t
ap ap t x0 x1   =   x0
ap ap t 1 5   =   1
ap ap t t i   =   t
ap ap t t ap inc 5   =   t
ap ap t ap inc 5 t   =   6
...
```

CHAPTER

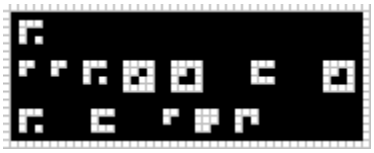
TWENTYFOUR

#22. FALSE

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

24.1 Image

This image was produced from the twenty-second radio transmission using *previously contributed code*.



24.2 Interpretation

Decoded as `f`, because it has a meaning of boolean False.

24.3 Decoded

```
f
ap ap f x0 x1   =   x1
f   =   ap s t
```

CHAPTER

TWENTYFIVE

#23. POWER OF TWO

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

25.1 Image

This image was produced from the twenty-third radio transmission using *previously contributed code*.

25.2 Interpretation

Recursive function: `pwr2` definition uses `pwr2`.

25.3 Decoded

```
pwr2
pwr2   =   ap ap s ap ap c ap eq 0 1 ap ap b ap mul 2 ap ap b pwr2 ap add -1
ap pwr2 0   =   ap ap ap s ap ap c ap eq 0 1 ap ap b ap mul 2 ap ap b pwr2 ap add -1 0
ap pwr2 0   =   ap ap ap ap c ap eq 0 1 0 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 0
ap pwr2 0   =   ap ap ap ap eq 0 0 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 0
ap pwr2 0   =   ap ap t 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 0
ap pwr2 0   =   1
ap pwr2 1   =   ap ap ap s ap ap c ap eq 0 1 ap ap b ap mul 2 ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap ap ap c ap eq 0 1 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap ap ap eq 0 1 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap f 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap ap b pwr2 ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap pwr2 ap ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap ap s ap ap c ap eq 0 1 ap ap b ap mul 2 ap ap b␣
↪pwr2 ap add -1 ap ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap ap ap c ap eq 0 1 ap ap add -1 1 ap ap ap b ap mul␣
↪2 ap ap b pwr2 ap add -1 ap ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap ap ap eq 0 ap ap add -1 1 1 ap ap ap b ap mul 2 ap␣
↪ap b pwr2 ap add -1 ap ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap ap ap eq 0 0 1 ap ap ap b ap mul 2 ap ap b pwr2 ap␣
↪add -1 ap ap add -1 1
ap pwr2 1   =   ap ap mul 2 ap ap t 1 ap ap ap b ap mul 2 ap ap b pwr2 ap add -1 ap␣
↪ap add -1 1
ap pwr2 1   =   ap ap mul 2 1
ap pwr2 1   =   2
ap pwr2 2   =   ap ap ap s ap ap c ap eq 0 1 ap ap b ap mul 2 ap ap b pwr2 ap add -1 2
...
ap pwr2 2   =   4
ap pwr2 3   =   8
ap pwr2 4   =   16
ap pwr2 5   =   32
ap pwr2 6   =   64
ap pwr2 7   =   128
ap pwr2 8   =   256
...
```

CHAPTER

TWENTYSIX

#24. I COMBINATOR

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

26.1 Image

This image was produced from the twenty-fourth radio transmission using *previously contributed code*.

26.2 Interpretation

```
i(x) = x
```

26.3 Decoded

```
i
ap i x0    =    x0
ap i 1    =    1
ap i i    =    i
ap i add    =    add
ap i ap add 1    =    ap add 1
...
```

CHAPTER

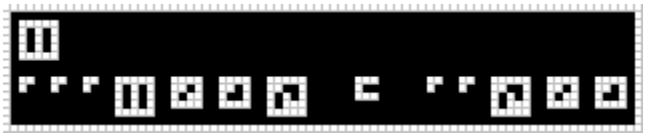
TWENTYSEVEN

#25. CONS (OR PAIR)

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

27.1 Image

This image was produced from the twenty-fifth radio transmission using *previously contributed code*.



27.2 Interpretation

27.3 Decoded

```
cons
ap ap ap cons x0 x1 x2   =   ap ap x2 x0 x1
```

CHAPTER

TWENTYEIGHT

#26. CAR (FIRST)

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

28.1 Image

This image was produced from the twenty-sixth radio transmission using *previously contributed code*.

28.2 Interpretation

28.3 Decoded

```
car
ap car ap ap cons x0 x1   =   x0
ap car x2   =   ap x2 t
```

CHAPTER

TWENTYNINE

#27. CDR (TAIL)

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

29.1 Image

This image was produced from the twenty-seventh radio transmission using *previously contributed code*.

29.2 Interpretation

29.3 Decoded

```
cdr
ap cdr ap ap cons x0 x1   =   x1
ap cdr x2   =   ap x2 f
```

CHAPTER

THIRTY

#28. NIL (EMPTY LIST)

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

30.1 Image

This image was produced from the twenty-eighth radio transmission using *previously contributed code*.

30.2 Interpretation

30.3 Decoded

```
nil
ap nil x0   =   t
```

CHAPTER

THIRTYONE

#29. IS NIL (IS EMPTY LIST)

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

31.1 Image

This image was produced from the twenty-ninth radio transmission using *previously contributed code*.

31.2 Interpretation

31.3 Decoded

```
isnil
ap isnil nil   =   t
ap isnil ap ap cons x0 x1   =   f
```

CHAPTER

THIRTYTWO

#30. LIST CONSTRUCTION SYNTAX

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

32.1 Image

This image was produced from the thirtieth radio transmission using *previously contributed code*.

32.2 Interpretation

32.3 Decoded

```
( , )
( )   =   nil
( x0 )   =   ap ap cons x0 nil
( x0 , x1 )   =   ap ap cons x0 ap ap cons x1 nil
( x0 , x1 , x2 )   =   ap ap cons x0 ap ap cons x1 ap ap cons x2 nil
( x0 , x1 , x2 , x5 )   =   ap ap cons x0 ap ap cons x1 ap ap cons x2 ap ap cons x5␣
↪nil
...
```

CHAPTER

THIRTYTHREE

#31. VECTOR

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

33.1 Image

This image was produced from the thirty-first radio transmission using *previously contributed code*.

33.2 Interpretation

Alias for `cons` that looks nice in "vector" usage context.

33.3 Decoded

```
vec
vec   =   cons
```

CHAPTER

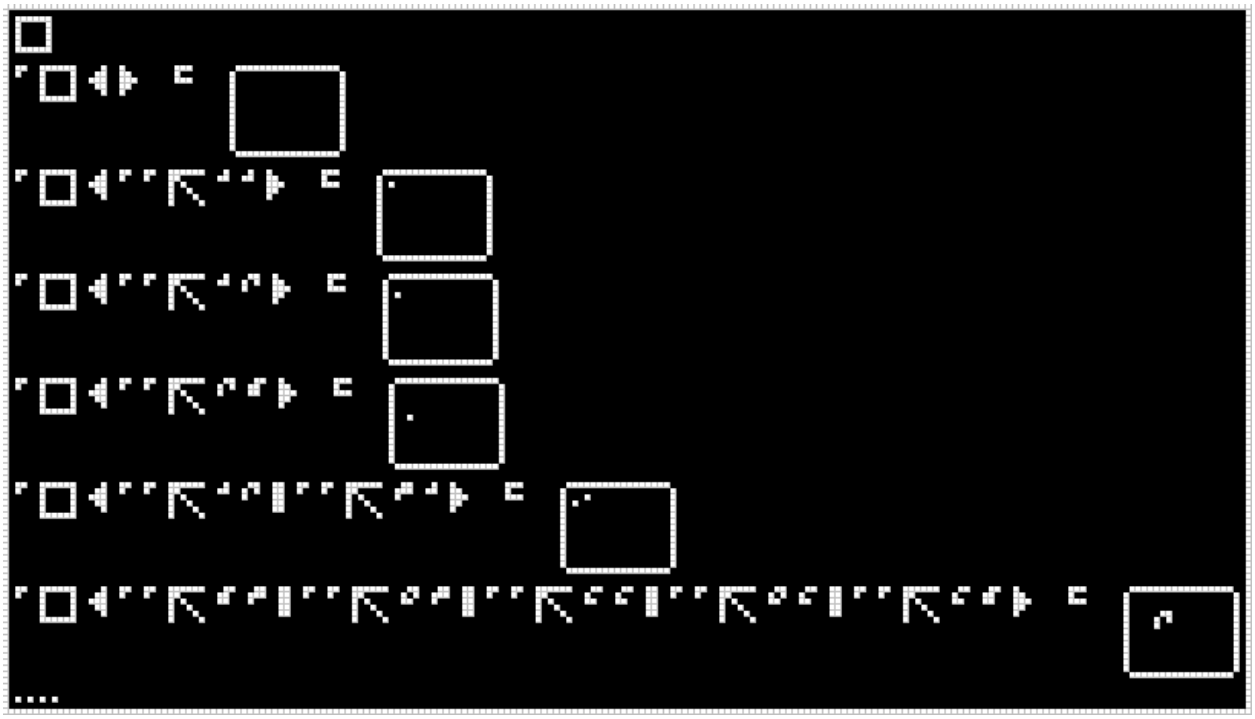
THIRTYFOUR

#32. DRAW

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

34.1 Image

This image was produced from the thirty-second radio transmission using *previously contributed code*.



34.2 Interpretation

It draws a list of coordinates as dots on a picture.

34.3 Decoded

```
draw
ap draw ( )   =   |picture1|
ap draw ( ap ap vec 1 1 )   =   |picture2|
ap draw ( ap ap vec 1 2 )   =   |picture3|
ap draw ( ap ap vec 2 5 )   =   |picture4|
ap draw ( ap ap vec 1 2 , ap ap vec 3 1 )   =   |picture5|
ap draw ( ap ap vec 5 3 , ap ap vec 6 3 , ap ap vec 4 4 , ap ap vec 6 4 , ap ap vec 4␣
↪5 )   =   |picture6|
...
```

CHAPTER

THIRTYFIVE

#33. CHECKERBOARD

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

35.1 Image

This image was produced from the thirty-third radio transmission using *previously contributed code*.

35.2 Interpretation

Draws a checkerboard of the specified size.

35.3 Decoded

```
checkerboard
checkerboard = ap ap s ap ap b s ap ap c ap ap b c ap ap b ap c ap c ap ap s ap ap b␣
↪s ap ap b ap b ap ap s i i lt eq ap ap s mul i nil ap ap s ap ap b s ap ap b ap b␣
↪cons ap ap s ap ap b s ap ap b ap b cons ap c div ap c ap ap s ap ap b b ap ap c ap␣
↪ap b b add neg ap ap b ap s mul div ap ap c ap ap b b checkerboard ap ap c add 2
ap ap checkerboard 7 0   =   |picture1|
ap ap checkerboard 13 0   =   |picture2|
```

CHAPTER

THIRTYSIX

#34. MULTIPLE DRAW

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

36.1 Image

This image was produced from the thirty-fourth radio transmission using *previously contributed code*.

36.2 Interpretation

Takes a list of lists of 2D-points and returns a list of rendered pictures.

It applies draw function to all items of the list.

36.3 Decoded

```
multipledraw
ap multipledraw nil    =    nil
ap multipledraw ap ap cons x0 x1    =    ap ap cons ap draw x0 ap multipledraw x1
```

CHAPTER

THIRTYSEVEN

#35. MODULATE LIST

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

37.1 Image

This image was produced from the thirty-fifth radio transmission using *previously contributed code*.

37.2 Interpretation

Apply *#13. Modulate* to a list constructed with *#25. Cons (or Pair)* or *#30. List Construction Syntax*.

37.3 Decoded

```
mod cons
ap mod nil   =   [nil]
ap mod ap ap cons nil nil   =   [ap ap cons nil nil]
ap mod ap ap cons 0 nil   =   [ap ap cons 0 nil]
ap mod ap ap cons 1 2   =   [ap ap cons 1 2]
ap mod ap ap cons 1 ap ap cons 2 nil   =   [ap ap cons 1 ap ap cons 2 nil]
ap mod ( 1 , 2 )   =   [( 1 , 2 )]
ap mod ( 1 , ( 2 , 3 ) , 4 )   =   [( 1 , ( 2 , 3 ) , 4 )]
...
```

CHAPTER

THIRTYEIGHT

#36. SEND (0)

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

38.1 Image

This image was produced from the thirty-sixth radio transmission using *previously contributed code*.

38.2 Interpretation

:1678847 is decreasing over time at a rate of 1/3 per second and will reach 0 at the icfp contest main round deadline.

38.3 Decoded

```
:1678847
ap send ( 0 )    =    ( 1 , :1678847 )
```

CHAPTER

THIRTYNINE

#37. IS 0

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

39.1 Image

This image was produced from the thirty-seventh radio transmission using *previously contributed code*.

39.2 Interpretation

Function `if0` compares the first argument to 0 and picks the second argument if equal, else third.

39.3 Decoded

```
if0
ap ap ap if0 0 x0 x1   =   x0
ap ap ap if0 1 x0 x1   =   x1
```

CHAPTER

FORTY

#38. INTERACT

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

40.1 Image

This image was produced from the thirty-eighth radio transmission using *previously contributed code*.

40.2 Interpretation

Is a function that takes an "interaction-protocol", some data (maybe "protocol" dependent), and some pixel. It returns some new data, and a list of pictures.

Note that during the execution it sometimes uses the *send* function to communicate with spacecraft.

```
// list function call notation
f38 protocol (flag, newState, data) = if flag == 0
                then (modem newState, multipledraw data)
                else interact protocol (modem newState) (send data)
interact protocol state vector = f38 protocol (protocol state vector)



// mathematical function call notation
f38(protocol, (flag, newState, data)) = if flag == 0
                then (modem(newState), multipledraw(data))
                else interact(protocol, modem(newState), send(data))
interact(protocol, state, vector) = f38(protocol, protocol(state, vector))
```

`mod` is defined on `cons`, `nil` and `numbers` only. So `modem` function seems to be the way to say that it's argument consists of numbers and lists only.

So we can assume that `newState` is always list of list of ... of numbers.

After experimenting with the galaxy interaction protocol we have found out several good ideas:

1. We can choose any `vector` to pass it to the interact function. But a convenient way to input this vectors — is clicking on the image pixel from the previous interact execution result.
3. We need to draw the images passed to multipledraw somehow. A convenient way to do it — is to overlay images one over another using different colors for different images.

40.3 Decoded

```
interact
ap modem x0 = ap dem ap mod x0
ap ap f38 x2 x0 = ap ap ap if0 ap car x0 ( ap modem ap car ap cdr x0 , ap␣
↪multipledraw ap car ap cdr ap cdr x0 ) ap ap ap interact x2 ap modem ap car ap cdr␣
↪x0 ap send ap car ap cdr ap cdr x0
ap ap ap interact x2 x4 x3 = ap ap f38 x2 ap ap x2 x4 x3
```

CHAPTER

FORTYONE

#39. INTERACTION PROTOCOL

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

41.1 Image

This image was produced from the thirty-ninth radio transmission using *previously contributed code*.

41.2 Interpretation

Start the protocol passing `nil` as the initial state and `(0, 0)` as the initial point. Then iterate the protocol passing new points along with states obtained from the previous execution.

41.3 Decoded

```
interact
ap ap ap interact x0 nil ap ap vec 0 0 = ( x16 , ap multipledraw x64 )
ap ap ap interact x0 x16 ap ap vec x1 x2 = ( x17 , ap multipledraw x65 )
ap ap ap interact x0 x17 ap ap vec x3 x4 = ( x18 , ap multipledraw x66 )
ap ap ap interact x0 x18 ap ap vec x5 x6 = ( x19 , ap multipledraw x67 )
...
```

CHAPTER

FORTYTWO

#40. STATELESS DRAWING PROTOCOL

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

42.1 Image

This image was produced from the fortieth radio transmission using *previously contributed code*.

42.2 Interpretation

42.3 Decoded

```
ap interact statelessdraw
ap ap statelessdraw nil x1 = ( 0 , nil , ( ( x1 ) ) )
statelessdraw = ap ap c ap ap b b ap ap b ap b ap cons 0 ap ap c ap ap b b cons ap ap␣
↪c cons nil ap ap c ap ap b cons ap ap c cons nil nil
ap ap ap interact statelessdraw nil ap ap vec 1 0 = ( nil , ( [1,0] ) )
ap ap ap interact statelessdraw nil ap ap vec 2 3 = ( nil , ( [2,3] ) )
ap ap ap interact statelessdraw nil ap ap vec 4 1 = ( nil , ( [4,1] ) )
...
```

CHAPTER

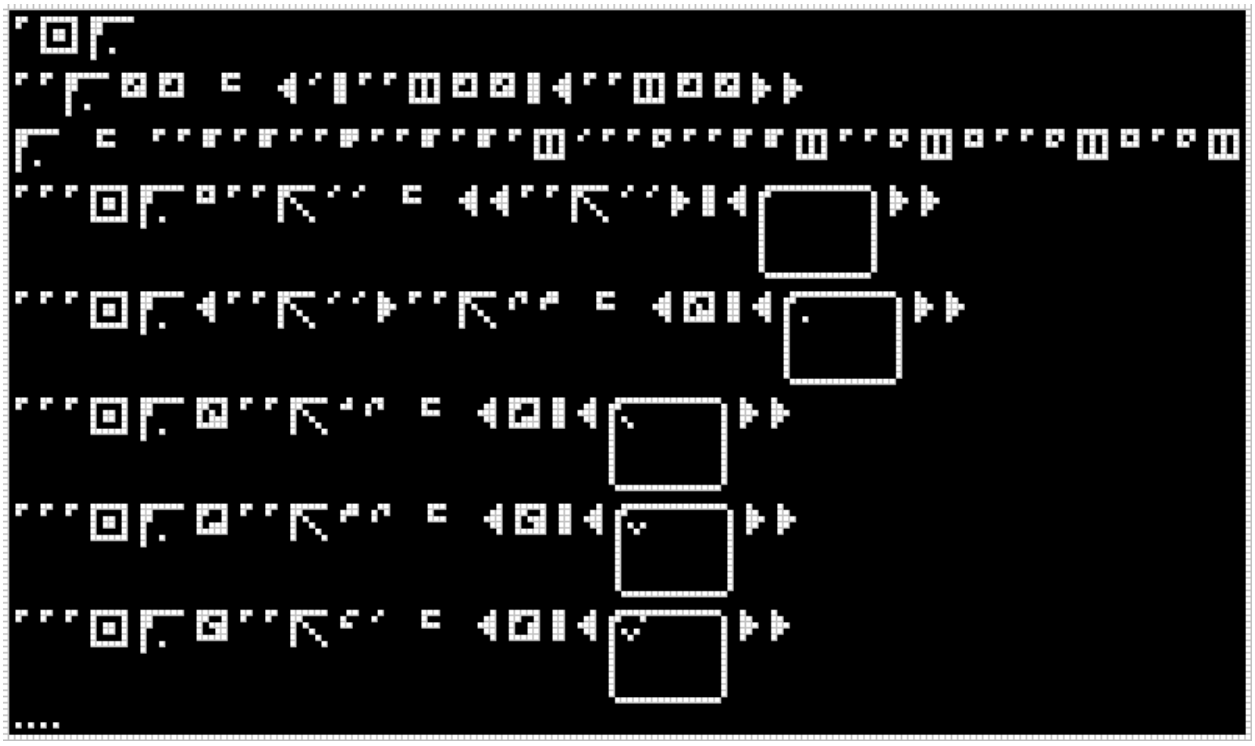
FORTYTHREE

#41. STATEFUL DRAWING PROTOCOL

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

43.1 Image

This image was produced from the forty-first radio transmission using *previously contributed code*.



43.2 Interpretation

It gives us back the variable bound to the draw state, so we can set the next pixel with the next call.

43.3 Decoded

```
ap interact :67108929
ap ap :67108929 x0 x1 = ( 0 , ap ap cons x1 x0 , ( ap ap cons x1 x0 ) )
:67108929 = ap ap b ap b ap ap s ap ap b ap b ap cons 0 ap ap c ap ap b b cons ap ap␣
↪c cons nil ap ap c cons nil ap c cons
ap ap ap interact :67108929 nil ap ap vec 0 0 = ( ( ap ap vec 0 0 ) , ( [0,0] ) )
ap ap ap interact :67108929 ( ap ap vec 0 0 ) ap ap vec 2 3 = ( x2 , ( [0,0;2,3] ) )
ap ap ap interact :67108929 x2 ap ap vec 1 2 = ( x3 , ( [0,0;2,3;1,2] ) )
ap ap ap interact :67108929 x3 ap ap vec 3 2 = ( x4 , ( [0,0;2,3;1,2;3,2] ) )
ap ap ap interact :67108929 x4 ap ap vec 4 0 = ( x5 , ( [0,0;2,3;1,2;3,2;4,0] ) )
...
```

CHAPTER

FORTYFOUR

#42. GALAXY

Note: Following documentation is a cooperative result combined from our Discord chat. Thanks to everyone who helped!

44.1 Image

This image was produced from the forty-second radio transmission using *previously contributed code*.

44.2 Interpretation

We believe that this message tells us to run an *interaction protocol* called `galaxy`. This protocol is defined in the last line of the `huge message` included on this page.

Messages *#38* and *#39* describe how to run a protocol. As we can see from *message #38*, a protocol takes a *vector* and returns a list of *pictures*.

Messages *#40* and *#41* define two simple protocols and demonstrate their behavior during execution.

44.3 Huge Transmission Text

`Download` the textual representation of the large transmission.

44.4 Decoded

```
ap interact galaxy = ...
```

CHAPTER

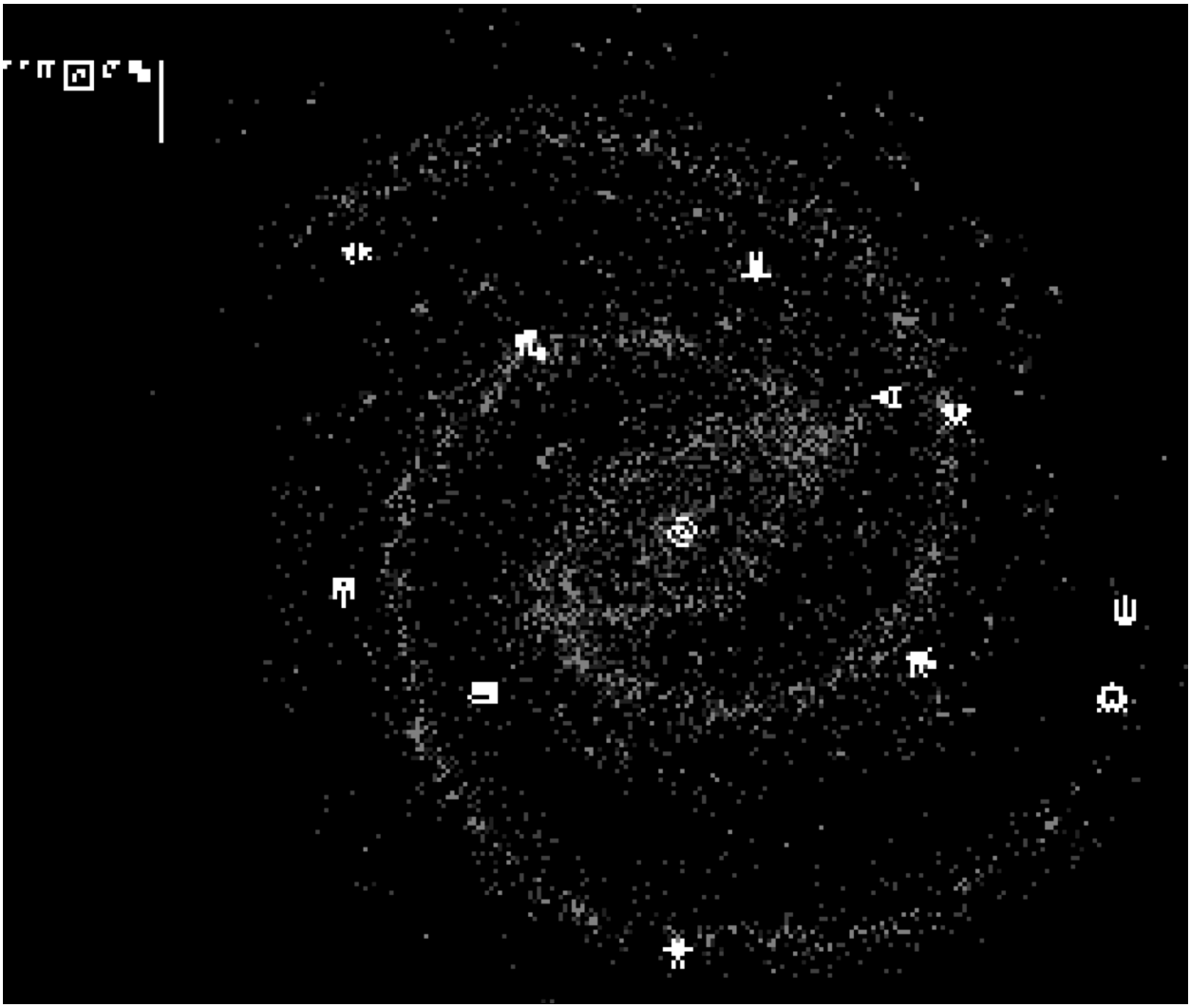
FORTYFIVE

FINAL TOURNAMENT

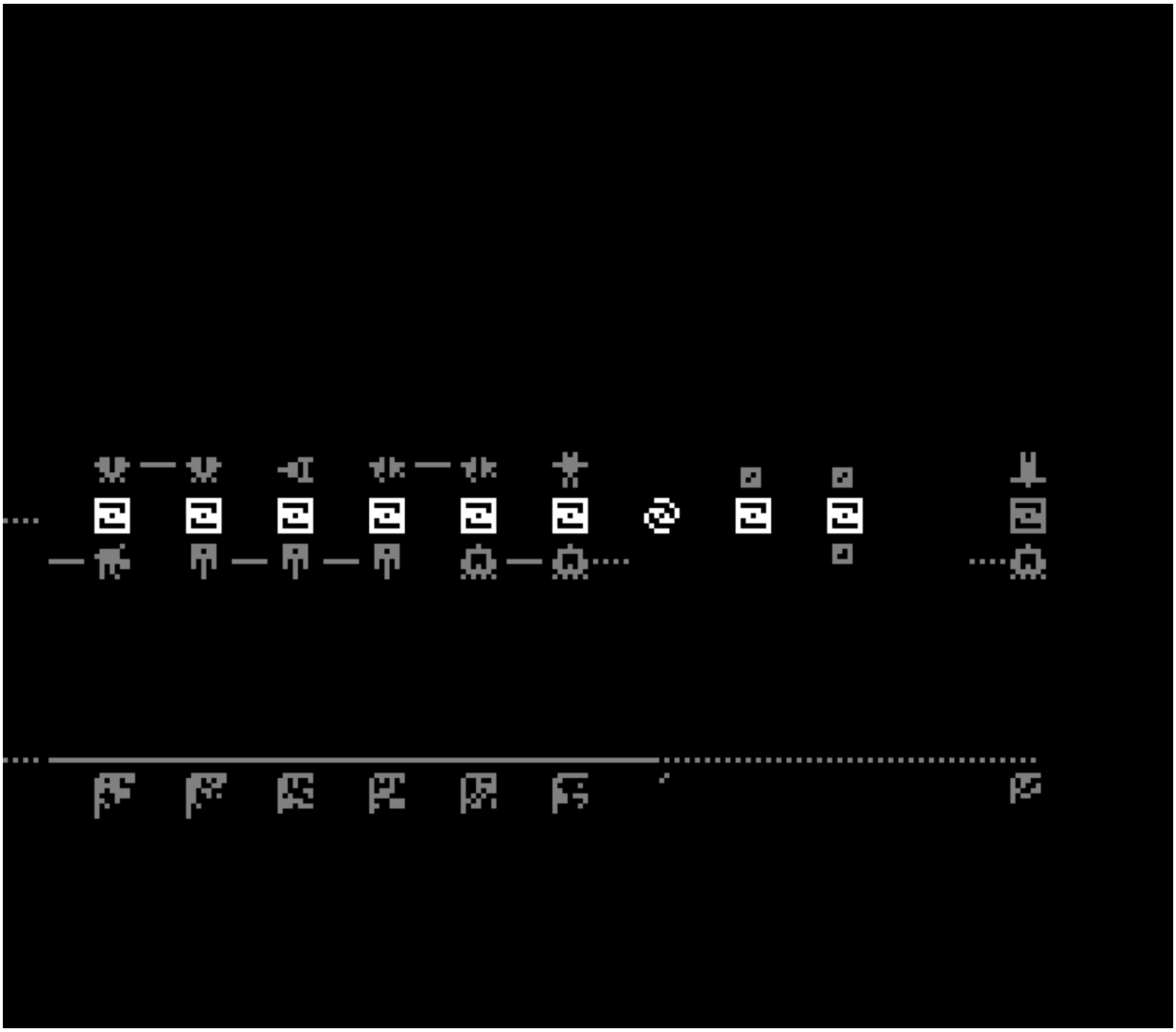
45.1 What We Know So Far

We continued to explore our "galaxy evaluator" and here's the summary of our findings.

We were demonstrated a historical perspective of lots of civilisations (including us!) spread throughout our Galaxy.



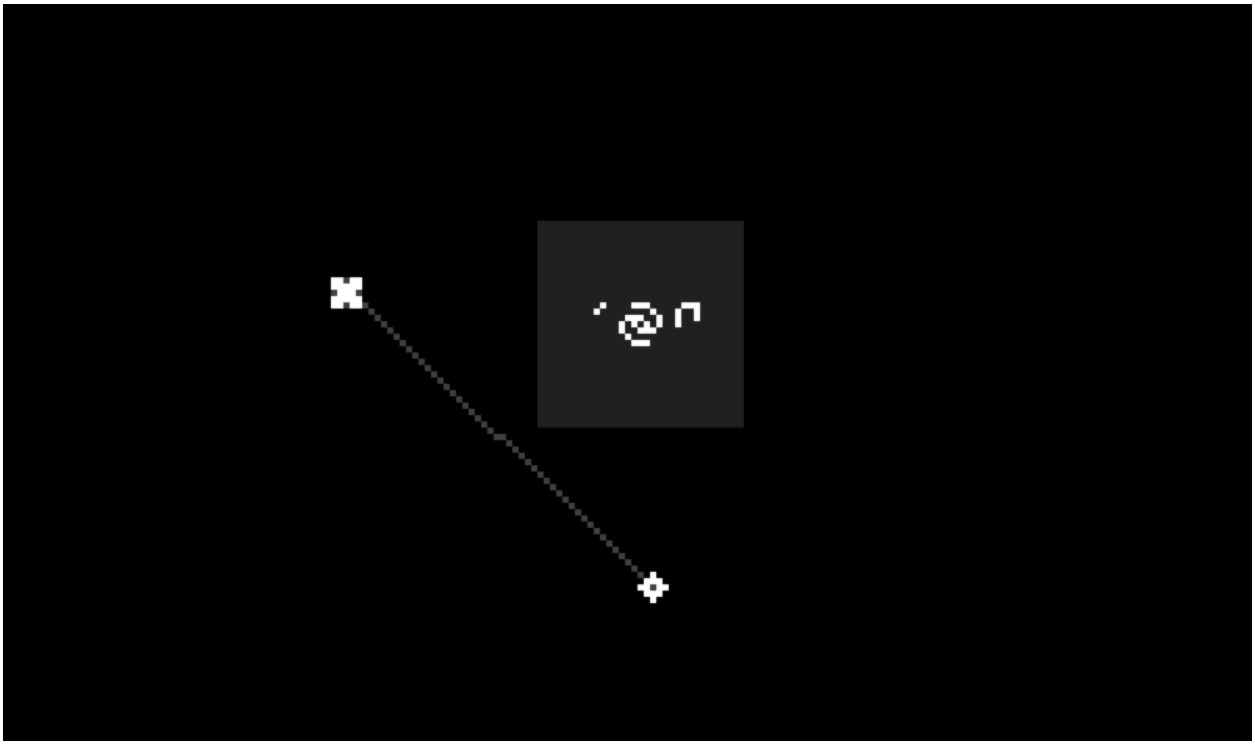
We saw an intergalactic tournament among the civilisations that lasts for ages.



The numbers at the bottom of this screen indicate that on July 20 at 13:00 UTC (a wild guess!) there will be the final battle between an alien race and the humankind. What happens if the humankind loses? We don't know (yet).

We could also watch the "replays" of the past battles. Then we were able to explore what looks like a series of "tutorial levels" demonstrating the basic concepts of space combat. No doubt we can use these materials to learn how to fight.

We were able to explore what a "battle" is. One (or many) "ships" seem to "orbit" a strange square "planet".



Also, these “ships” bear one of two responsibilities: they are either “attacking ships” or “defending ships”. In order to win a “battle”, “attackers” have to destroy other ships in a fixed number of “turns”. Unlike “attackers”, “defenders” win a “battle” if they are not destroyed in a fixed number of “turns”. Note that we don’t know if a “draw” is a possible result of a “battle”.

Although we were unable to explore all “tutorial levels” and get through them, we’ve made further progress. We noticed that the “states” (see the *message #42*) of the “evaluator” are nearly identical after every “battle” that we have explored. The only difference is that a single number keeps incrementing. We’ve tried our luck and put a larger number in there—and that was the right move.

We believe that we’ve managed to enter the “multiplayer mode” which can be used to fight other players. To find the

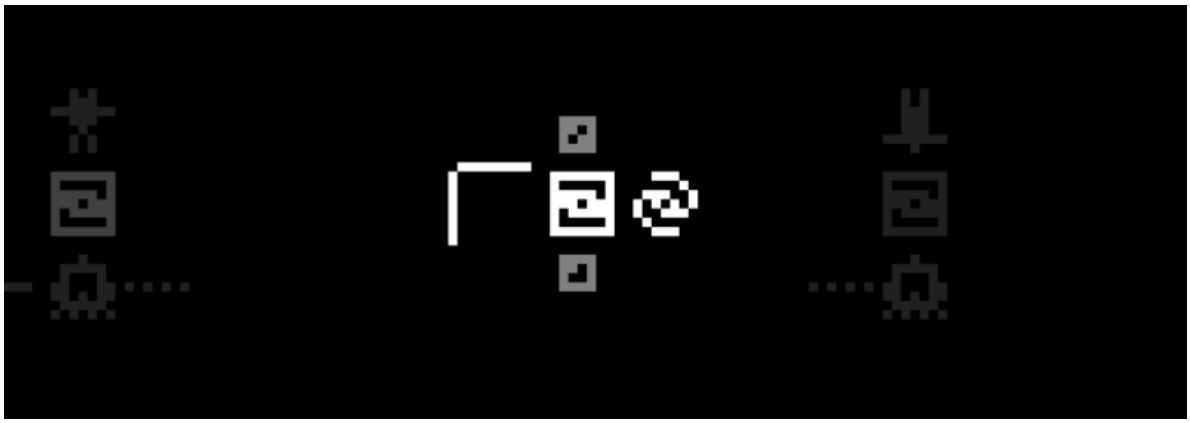
best candidate to fight for the humankind in the final battle, we are going to set up our own local tournament using this "multiplayer mode". We will accept submissions for this tournament before the final battle countdown ends.

45.2 How to Play the Local Tournament

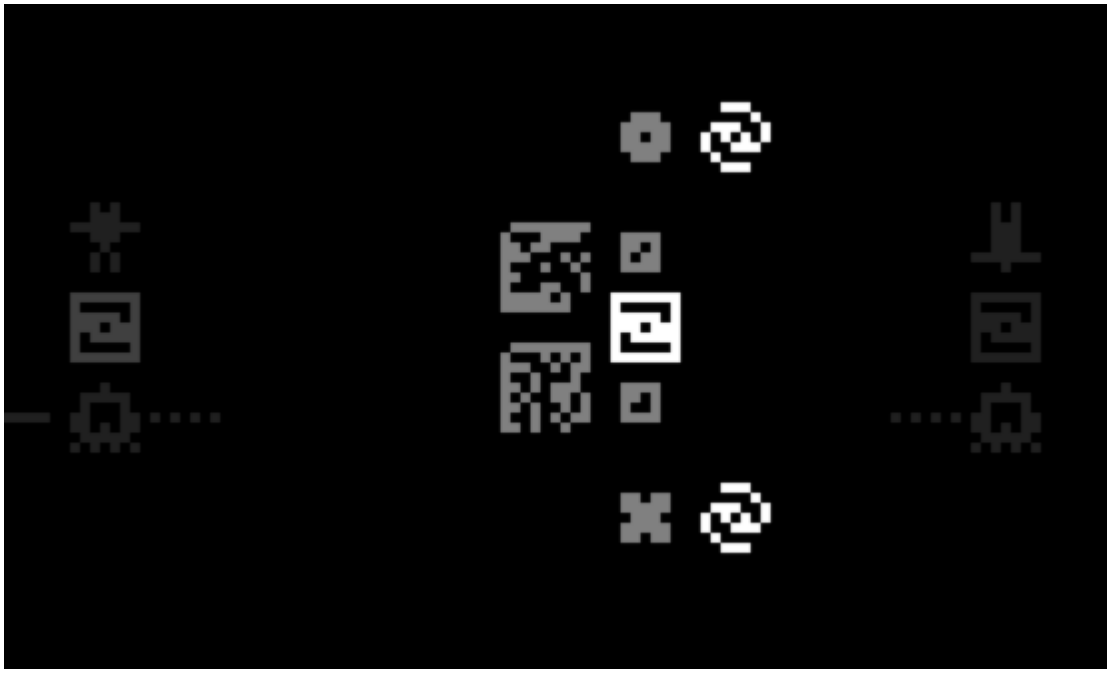
This section describes the way we run your submitted code on our server. Your submission can play games using its internal Galaxy Pad UI instance or, alternatively, via direct proxy calls.

We run some preparation steps for you, so you don't have to do it:

1. In our internal Galaxy Pad instance we create a new game and generate player keys for both attacker and defender. First, we click on the two-player game button:

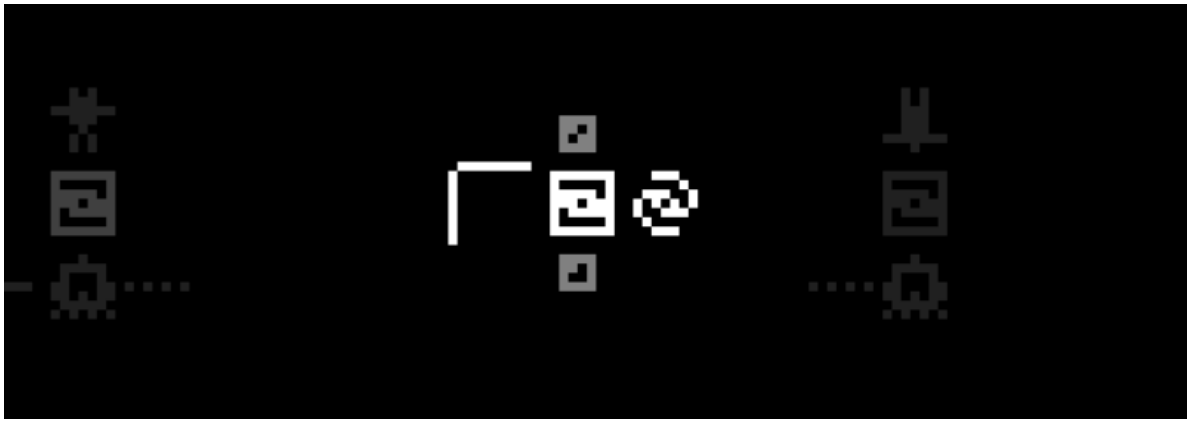
   

   Then, click on the Galaxy button:

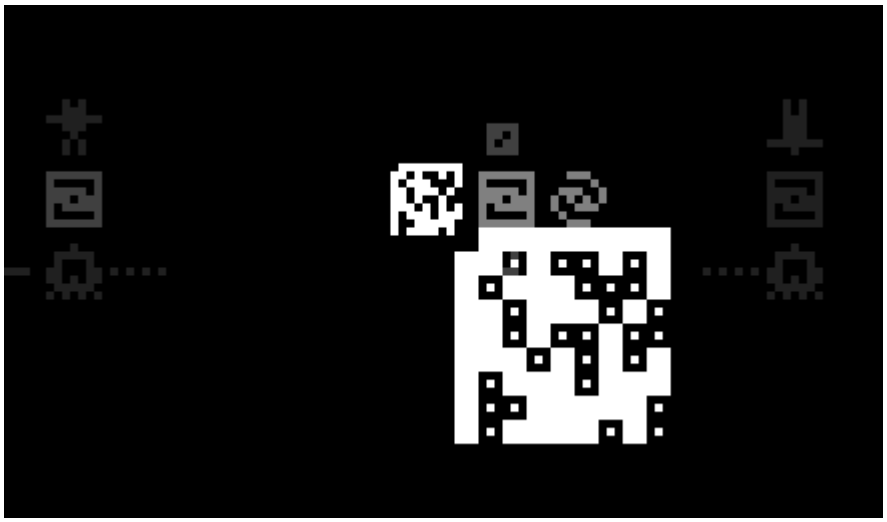
   

2. We run your submission container with `serverUrl` and `playerKey` as command line arguments. Note that you **must** use provided `serverUrl` as a base URL for all outgoing `aliens/send` requests to alien proxy. For example, `serverUrl` can be `http://server:12345`. In this case you should send requests to `http://server:12345/aliens/send`.

3. Your bot **must** join the game using the provided `playerKey`. Your bot can create its own internal Galaxy Pad instance and do it via the UI. First, click on the two-player game button:



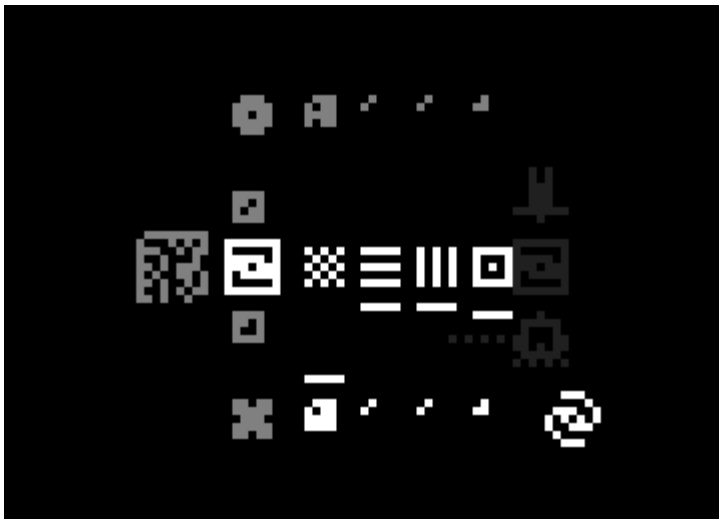
Then, click on the Player Key button on the left and input player key pixel by pixel (you can hack the state to avoid manual input):



Then, click on the top-left Player Key button to confirm joining the game.

Alternatively your bot can do it without the Galaxy Pad using our HTTP proxy directly via `JOIN` request (see below).

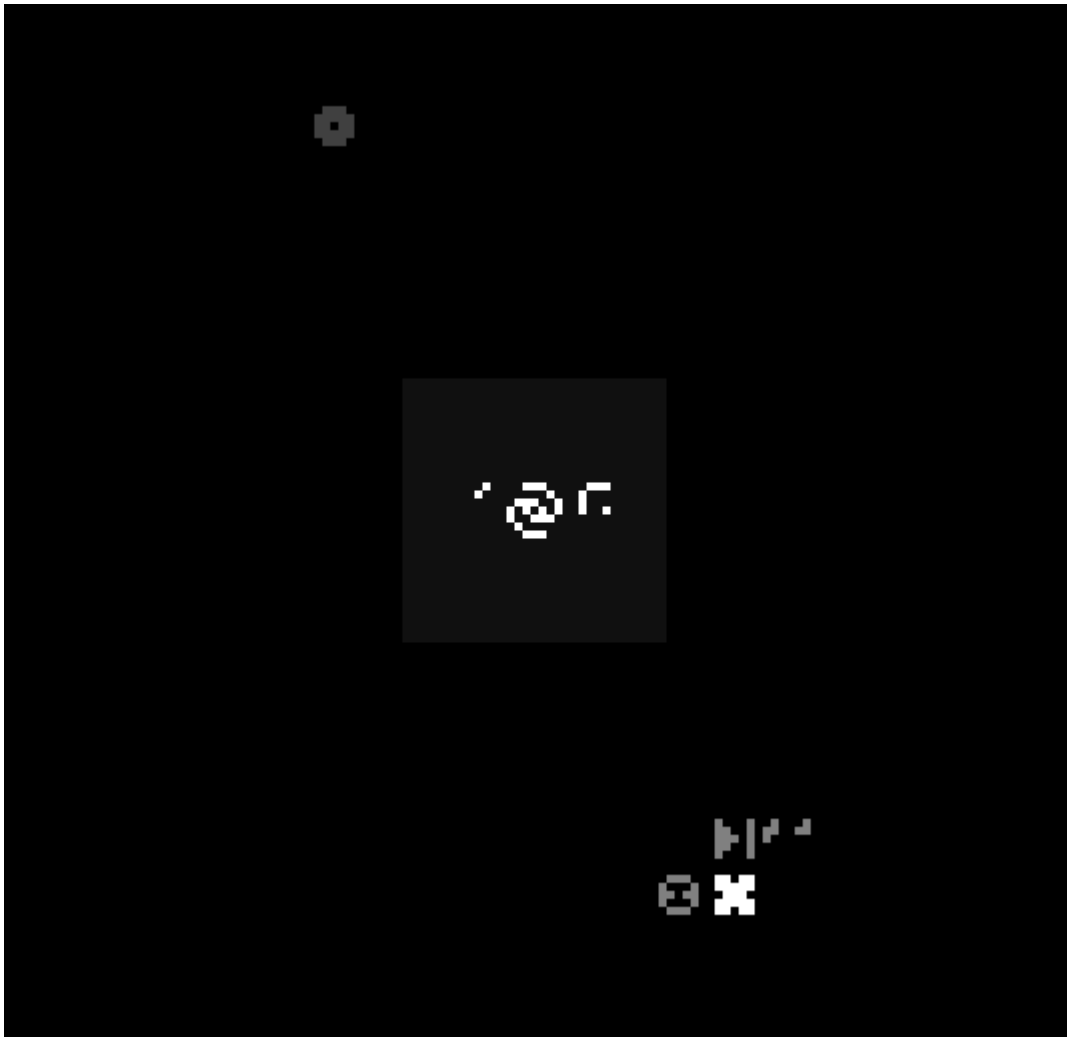
4. Your bot **must** choose your initial ship parameters and start playing after successfuly joining. Your bot can do it via your internal Galaxy Pad UI:



Then, click on the Galaxy button to start the game.

Alternatively your bot can do it without the Galaxy Pad using our HTTP proxy directly via `START` request (see below).

5. Your bot **must** issue commands for your ships until the game is finished. Your bot can do it via your internal Galaxy Pad UI as you did in the tutorials:



Clicking on the Galaxy button sends all your selected commands to the alien proxy.

Alternatively your bot can do it without the Galaxy Pad using our HTTP proxy directly via `COMMANDS` request (see below).

45.3 Timeouts

Each action required from your bot **must** be done fast enough, or we will disconnect it from the game and give the victory to your opponent.

- `JOIN`: you should send it within **10 seconds** after we run your `run.sh`.
- `START`: you should send it within **1 second** after you receive the response to `JOIN`.
- `COMMANDS`: you should send it within **1 second** after you receive the response to previous `START` or `COMMANDS`.

Also you have a timeout for the entire game (all `COMMANDS`, but not `JOIN` and `START`): **2 minute** total. We know that a game runs for a maximum of **384** turns, so it's up to you how to use this time.

45.4 Implementation Details

Of course, you can just run your Galaxy Pad and emulate clicks on it.

But we also have partially reverse-engineered the protocol so you can use this knowledge to send requests to the Alien Proxy directly, without running your Galaxy Pad at all.

Here is a pseudo code:

```
main (args)
{
    // parse command line arguments
    serverUrl = args[0]
    playerKey = args[1]

    // make valid JOIN request using the provided playerKey
    joinRequest = makeJoinRequest(playerKey)

    // send it to aliens and get the GameResponse
    gameResponse = send(serverUrl, joinRequest)

    // make valid START request using the provided playerKey and gameResponse␣
↪returned from JOIN
    startRequest = makeStartRequest(playerKey, gameResponse)

    // send it to aliens and get the updated GameResponse
    gameResponse = send(serverUrl, startRequest)

    while (true) // todo: you MAY detect somehow that game is finished using␣
↪gameResponse
    {
        // make valid COMMANDS request using the provided playerKey and gameResponse␣
↪returned from START or previous COMMANDS
        commandsRequest = makeCommandsRequest(playerKey, gameResponse)

        // send it to aliens and get the updated GameResponse
        gameResponse = send(serverUrl, commandsRequest)
    }
}
```

45.5 Protocol

We denote unknown data as `xi` below.

45.5.1 CREATE

Note: You shouldn't call CREATE in your submissions. We do that for you. See *Implementation Details*.

One can use this request to create the new `playerKeys` to use them in the JOIN request.

```
( 1, 0 )
```

Response to that request has format:

```
(1, ((0, attackPlayerKey), (1, defenderPlayerKey)))
```

45.5.2 JOIN

```
(2, playerKey, (...unknown list...))
```

Purpose of the third item of this list is still unclear for us and we saw only empty list (`nil`) here. Maybe you will discover more and use it...

Response is described in the *GameResponse section*.

45.5.3 START

```
(3, playerKey, (x0, x1, x2, x3))
```

The third item of this list is always a list of 4 numbers – it's the initial ship parameters.

We noticed, that START doesn't finish successfully when `x3` is 0 or `xi`'s are too large.

Response is described in the *GameResponse section*.

45.5.4 COMMANDS

```
(4, playerKey, commands)
```

`commands` is the list of issued commands. Each item has format `(type, shipId, ...)`, where `...` denotes command-specific parameters. Some types of commands are described below.

Response is described in the *GameResponse section*.

Accelerate command

```
(0, shipId, vector)
```

Accelerates ship identified by `shipId` to the direction opposite to `vector`.

Detonate command

```
(1, shipId)
```

Detonates ship identified by `shipId`.

Shoot command

```
(2, shipId, target, x3)
```

`target` is a vector with coordinates of the shooting target.

45.5.5 GameResponse

In the case of wrong request:

```
(0)
```

In case of correct request:

```
(1, gameStage, staticGameInfo, gameState)
```

- `1` indicates success
- `gameStage` is a number
  - `0` indicates that the game has not started yet
  - `1` indicates that the game has already started
  - `2` indicates that the game has finished
- `staticGameInfo` doesn't change from turn to turn during the whole game
- `gameState` changes from turn to turn

```
staticGameInfo = (x0, role, x2, x3, x4)
```

`role`

- `0` indicates that you are in the attacker role
- `1` indicates that you are in the defender role

```
gameState = (gameTick, x1, shipsAndCommands)
```

- `gameTick` is the time inside the game
- `shipsAndCommands` is a list of items, each item has a structure of `(ship, appliedCommands)`
  - `appliedCommands` is a list of commands applied to the `ship` on the previous tick
  - `ship` is the ship state description

```
ship = (role, shipId, position, velocity, x4, x5, x6, x7)
```

- `position` is a vector with the ship coordinates
- `velocity` is a vector with the ship velocity

45.6 Scoring

Local tournament consists of several stages. Each stage has a hard deadline:

1. 24 hours before the Alien Deadline (not scored, see below)
2. 18 hours before the Alien Deadline
3. 12 hours before the Alien Deadline
4. 9 hours before the Alien Deadline
5. 6 hours before the Alien Deadline
6. 4 hours before the Alien Deadline
7. 2 hours before the Alien Deadline (leaderboard frozen)
8. Alien Deadline (July 20 at 13:00 UTC)

Teams submit their solutions as described in the submission system documentation. A team must select **only one** built and tested submission as their **active** submission selected for rating games. This choice is made via the Submissions page (click on a row to select). Note that new commits **do not** automatically become active unless there is a `#release` word in the commit message.

Before letting team's submission participate in the tournament our system will test the submission's ability to join and start a game as an attacker and as a defender. It means that vanilla starter kits are **no longer considered valid submissions**.

At the end of each stage our system will stop accepting new submissions for that stage. It means that your **active** submission at the end of the stage becomes your **final** submission for that stage.

Our tournament system uses TrueSkill rating system to pair opponents and rank submissions in each stage. After the end of each stage our system will run additional rounds of games until all the TrueSkill ratings settle. Then we will assign score to top 50 submissions according to a formula:

$$score = \lfloor 50^{(50-rank)/50} \rfloor$$

... where `rank` is zero-based position in the leaderboard for this stage.

Total score for a team is the sum of the scores of that team **for stages 2..8**. The first stage earns no score and serves to make you familiar with the system.

After the final Alien Deadline we will stop accepting new submissions entirely. We will trigger one final build in all submission branches in all repositories exactly at the moment of the deadline. Please note that we build only the latest commit in each branch. If you push N commits at once, it will result in only one built submission. Our build process is not instant: the checkout can happen several minutes after the deadline. Please don't push anything you don't want to submit after the deadline.

As soon as all the submissions are built and tested, we'll make an announcement. After this announcement you will have **exactly 30 minutes** to select your **final submission for the whole tournament**. This final submission will play in the stage 8 until all the TrueSkill ratings settle.

Then we will run an **additional tournament stage** between the top 20 teams ranked by the **total score** earned in stages 2..8. Your final submission for the whole tournament will play this additional stage. Winners of this additional stage will fight against the aliens for the honor of the humankind. And declared as winners of the ICFP Contest 2020, of course.

Results of this additional stage and the whole contest will be made public at the ICFP 2020 in August 2020.

CHAPTER

FORTYSIX

GALAXY EVALUATOR IN PSEUDOCODE

```
// See video course https://icfpcontest2020.github.io/#/post/2054
class Expr
    optional Expr Evaluated

class Atom extends Expr
    string Name

class Ap extends Expr
    Expr Fun
    Expr Arg

class Vect
    number X
    number Y

Expr cons = Atom("cons")
Expr t = Atom("t")
Expr f = Atom("f")
Expr nil = Atom("nil")

Map<string, Expr> functions = PARSE_FUNCTIONS("galaxy.txt")

// See https://message-from-space.readthedocs.io/en/latest/message39.html
Expr state = nil
Vect vector = Vect(0, 0)

while(true)
    Expr click = Ap(Ap(cons, Atom(vector.X)), Atom(vector.Y))
    var (newState, images) = interact(state, click)
    PRINT_IMAGES(images)
    vector = REQUEST_CLICK_FROM_USER()
    state = newState

// See https://message-from-space.readthedocs.io/en/latest/message38.html
(Expr, Expr) interact(Expr state, Expr event)
    Expr expr = Ap(Ap(Atom("galaxy"), state), event)
    Expr res = eval(expr)
    // Note: res will be modulatable here (consists of cons, nil and numbers only)
    var [flag, newState, data] = GET_LIST_ITEMS_FROM_EXPR(res)
    if (asNum(flag) == 0)
        return (newState, data)
    return interact(newState, SEND_TO_ALIEN_PROXY(data))

Expr eval(Expr expr)
```

(continues on next page)

(continued from previous page)

```
    if (expr.Evaluated != null)
        return expr.Evaluated
    Expr initialExpr = expr
    while (true)
        Expr result = tryEval(expr)
        if (result == expr)
            initialExpr.Evaluated = result
            return result
        expr = result

Expr tryEval(Expr expr)
    if (expr.Evaluated != null)
        return expr.Evaluated
    if (expr is Atom && functions[expr.Name] != null)
        return functions[expr.Name]
    if (expr is Ap)
        Expr fun = eval(expr.Fun)
        Expr x = expr.Arg
        if (fun is Atom)
            if (fun.Name == "neg") return Atom(-asNum(eval(x)))
            if (fun.Name == "i") return x
            if (fun.Name == "nil") return t
            if (fun.Name == "isnil") return Ap(x, Ap(t, Ap(t, f)))
            if (fun.Name == "car") return Ap(x, t)
            if (fun.Name == "cdr") return Ap(x, f)
        if (fun is Ap)
            Expr fun2 = eval(fun.Fun)
            Expr y = fun.Arg
            if (fun2 is Atom)
                if (fun2.Name == "t") return y
                if (fun2.Name == "f") return x
                if (fun2.Name == "add") return Atom(asNum(eval(x)) + asNum(eval(y)))
                if (fun2.Name == "mul") return Atom(asNum(eval(x)) * asNum(eval(y)))
                if (fun2.Name == "div") return Atom(asNum(eval(y)) / asNum(eval(x)))
                if (fun2.Name == "lt") return asNum(eval(y)) < asNum(eval(x)) ? t : f
                if (fun2.Name == "eq") return asNum(eval(x)) == asNum(eval(y)) ? t : f
                if (fun2.Name == "cons") return evalCons(y, x)
            if (fun2 is Ap)
                Expr fun3 = eval(fun2.Fun)
                Expr z = fun2.Arg
                if (fun3 is Atom)
                    if (fun3.Name == "s") return Ap(Ap(z, x), Ap(y, x))
                    if (fun3.Name == "c") return Ap(Ap(z, x), y)
                    if (fun3.Name == "b") return Ap(z, Ap(y, x))
                    if (fun3.Name == "cons") return Ap(Ap(x, z), y)
    return expr


Expr evalCons(Expr a, Expr b)
    Expr res = Ap(Ap(cons, eval(a)), eval(b))
    res.Evaluated = res
    return res

number asNum(Expr n)
    if (n is Atom)
        return PARSE_NUMBER(n.Name)
    ERROR("not a number")
```

CHAPTER

FORTYSEVEN

ALIEN PROXY PROTOCOL

Alien Proxy allows you to send requests to the spacecraft in orbit using our antenna.

It's a simple HTTP server.

47.1 Base Url

https://api.pegovka.space/

47.2 Send a Request to Spacecraft

Pass modulated string in the request body with a `Content-Type: text/plain` HTTP header.

Relative URL: `/aliens/send`

Sample request:

```
POST /aliens/send HTTP/1.1

1101000
```

47.2.1 Possible Responses

200 OK

You will get this response if the spacecraft responds fast enough.

Response body will contain modulated spacecraft response with a `Content-Type: text/plain` HTTP header.

Sample response:

```
HTTP/1.1 200 OK

1101100001110111110111101010101011100
```

302 Found

You will get this response if the spacecraft doesn't respond fast enough.

If the spacecraft doesn't respond fast enough we return `302 Found` status code. The `Location` response HTTP header will contain an URL where you can ask for the response again later. In fact, this header will always contain `/aliens/{responseId}`. It's a long-polling protocol, so you can make a new request to this location immediately after you got it. Many HTTP client implementations, e.g. C#'s `HttpClient`, can follow redirects automatically, so you don't deal with this.

Sample response:

```
HTTP/1.1 302 Found
Location: /aliens/75960227-653C-47E3-A47A-118A46AFFD4C
```

47.3 Get a Response From Spacecraft

Use this to get a response to the request you have sent earlier, in case the spacecraft didn't respond fast enough.

Relative URL: `/aliens/{responseId}`

Possible responses are the same as in `/aliens/send`.