Esoteric Programming Languages

blinry

17. Esoterischprogrammiernacht
Introduction

“esoteric” from Greek *esoterikos*, “belonging to an inner circle”
“esoteric” from Greek *esoterikos*, “belonging to an inner circle”

**Esoteric Programming Languages**
- proof-of-concept
- artistic expression
- challenge (for the designer and/or user)
- joke
“esoteric” from Greek *esoterikos*, “belonging to an inner circle”

**Esoteric Programming Languages**

- proof-of-concept
- artistic expression
- challenge (for the designer and/or user)
- joke

This presentation: Five Turing-complete languages
Brainfuck

- Designed by Urban Müller in 1993
Brainfuck

- Designed by Urban Müller in 1993
- Motivation: Small compiler (296 bytes)
Brainfuck

- Designed by Urban Müller in 1993
- Motivation: Small compiler (296 bytes)
- **Minimalist syntax**, only eight commands:
  > < + − , . [ ]
Brainfuck

- Designed by Urban Müller in 1993
- Motivation: Small compiler (296 bytes)
- **Minimalist syntax**, only eight commands:
  > < + - , . [ ]
- “brain fuck” = hard or complicated thing
Examples

1

```
>>>,------.<<<++++++++++.
```
# Examples

<table>
<thead>
<tr>
<th></th>
<th>Brainfuck</th>
<th>INTERCAL</th>
<th>Befunge</th>
<th>Malbolge</th>
<th>Shakespeare</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&gt;&gt;&gt;,------.&lt;&lt;&lt;+++++++++++</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example 1: >>>,------.<<<+++++++++++.

Input: f
Output: a

Example 2: ++++++[->+<]

Input: h
Output: 01101000
Examples

1. `>>> ,----- .<<<<+++++++ .`
Examples

1 >>>,-----..<<<++++++++++++.
Examples

1. `>>> ,----- .<><+----------+.`
Examples

1

>>>,----.<<<<+++.+
Examples

1. >>>,-----.<<<<<+>>>>+

Input: f
Output: a
Examples

1. `>>>,----.<<<<+++++++.`
Examples

1. >>>,------.<<<<...............
   Input: f
Examples

1. `>>>,------.<<<+++++++.`  
   Input: f  Output: a←
Examples

1. `>>>,------..<<+++++++++++.`  
   Input: f  
   Output: a←

1. `+++++[->+<]`
Examples

1. >>>,-----.<<<<<++.+++.+
   Input: f   Output: a←

1. ++++++ [->+<]
Examples

1. `>>>,------..<<<<+++
   Input: f   Output: a←`

2. `+++++
   [->+<]
   Input: h   Output: 01101000`
Examples

1. `>>>,------..<<<+++++++++++++.`
   - Input: f
   - Output: a ←

1. `+++++[->+<]`
### Examples

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;&gt;&gt;&gt;,------..&lt;&lt;&lt;&lt;&lt;+&lt;+++++++&quot;, Input: f</td>
<td>Output: a←</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>&quot;+++++[-&gt;+&lt;]&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Introduction**

**Brainfuck**

**INTERCAL**

**Befunge**

**Malbolge**

**Shakespeare**

**Conclusion**
Examples

1

```
>>>,------.<<<+++++++++++++
```

Input: f   Output: a←

1

```
+++++[->+<]
```

Input: h   Output: 01101000
# Introduction

Brainfuck
INTERCAL
Befunge
Malbolge
Shakespeare
Conclusion

## Examples

<table>
<thead>
<tr>
<th></th>
<th>Input: f</th>
<th>Output: a←</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&gt;&gt;&gt;,-----.&lt;&lt;&lt;+&lt;+&gt;+++</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>++++++[-&gt;+&lt;]</td>
<td></td>
</tr>
</tbody>
</table>
Examples

1. >>>,-------.<<<+++++.
   Input: f   Output: a←

1. +++++[->+<]
Examples

1
>>>,-------.<<<<<+++. 

1
+++++[->+<]

-]>[,<<<<[->+<]+[<+>-][-<]<<<<
2
<<<<<<+[<+][<<<<><+[<+]>+]<+[-]<
3
[]>.<[-]+[-]<>-]
Examples

1. `>>>,------.<<<+<+<+[->+<]>-]<<<<`  
   Input: f  Output: a←

2. `+++++[->+<]`  
   Input: h
Examples

1. `>>>,-------.<<<<+++.+++++++.`  
   Input: f  Output: a←

2. `+++++[->+<]`

   Input: h  Output: 01101000

4. `<<<<<<+[<+++++++++[->+++]+[>]>]`  
   `<]>.[-]>+-]`
Example: ASCII to binary

```
- 
[ 
  >,
  [ <<
    [-<] + [+->]+- >-
  ]
<<<<<
+[-]
+++<<++<<
[-] > +]
]```

Tape

```
... 0 0 0 0 0 0 0 0 0 0 0 0 0 ...```

Input

```
hello
```

Output

```
01101000
01100101
01101100
01101111
```
Example: ASCII to binary

```
- 
[ 
  >, 
  [ 
    << 
    [-<] 
    + 
    +[->+] [- 
    >> 
  ] 
  <<<<<<<< 
  +[- 
    <+++++++ 
    [->++++++<] >. 
    [-] 
    > 
    +] 
]
```

Tape

```
... 0 0 0 0 0 0 0 0 0 255 0 ...
```

Input

```
hello
```

Output

```
01101000
01100101
01101100
01101111
```
Example: ASCII to binary

```
- 
[ 
  >, 
  [ 
    << 
    [-><] 
    + 
    +[->+]- 
    >-
  ]
<<<<<<<<<
+[-
  <+++++++ 
  [->+++++[>.
  [-]
  >
  +]-
]
```

Tape
```
... 0 0 0 0 0 0 0 0 0 255 0 ... 
```

Input
```
hello
```

Output
```
01101000
01100101
01101100
01101111
```
Example: ASCII to binary

```
- [ >,< [ << [->]+- >- ]<<<<<<
+[-] <+++++++[-]>.
[-] > +][-]

Tape
... 0 0 0 0 0 0 0 0 255 0 ...

Input
hello

Output
```

```
1101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
- [ >, [ << [->-]+ >- ] ] <<<<<<<<+
  +[-<] +[->+]-
  >>-
```

Tape

```
... 0 0 0 0 0 0 0 0 255 104 ...
```

Input
```
ello
```

Output
Example: ASCII to binary

```
```

**Tape**

```
... 0 0 0 0 0 0 0 0 255 104 ...
```

**Input**

```
ello
```

**Output**

```
01101000
01100101
01101100
01101111
```
Example: ASCII to binary

```
-[
  >,
  [<<
    [-<]
    +
    +[->]+-
    >-
  ]
  <<<<<<<<
  +[-
    <+++++++]
    [->++++++<[>.
    [-]
    >
    +]-
]
```

Tape

```
... 0 0 0 0 0 0 0 0 255 104 ...
```

Input

```
ello
```

Output

```
01101000
01100101
01101100
01101111
```
Example: ASCII to binary

```
- [ >, [ << [-<] + [+->]- >- ] <<<<<<<<<<< +[-[+<++] [->+++++<]> [-] > +][-]
]```

**Tape**

```
... 0 0 0 0 0 0 0 0 255 104 ...
```

**Input**

```
ell0
```

**Output**

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
- [
  >,
  [<<[
  [-<]
  +
  +[->]+-
  >-
  ]
  <<<<<<<<
  +[-
    <+++++++[
    [-+++++++<].
    [-]
    >
    +]--
  ]]
```

Tape

```
... 0 0 0 0 0 0 0 0 255 104 ...
```

Input

```
ello
```

Output

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
1
2 -
3 [  
4   >,
5    [  
6      <<
7      [->]<
8      +  
9      +[->]+-  
10     >-
11 ]
12 <<<<<<<<<
13 +[-     
14    <+++++++ 
15     [->+++++[>].
16     [->     
17     >      
18     +]--
```

Tape

```
... 0 0 0 0 0 0 0 0 0 255 104 ...
```

Input

```
ello
```

Output

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
-  
[  
  >,  
  [  
    <<  
      [->]<  
      +  
      +[-]>+-  
      >=  
    ]  
  ]  
<<<<<<<<<<<  
+[-  
  <+++++++  
  [->++++++<]>  
  [-  
    >  
    +]--  
  ]  
]
```
Example: ASCII to binary

```
- [% >, [% << [% ->]+]- >] <<<<<<<<
```

**Tape**

```
... 0 0 0 0 0 0 0 2 255 104 ...
```

**Input**

```
ello
```

**Output**

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
- [
  >,
  [<
    [-<]
    +
    +[->]+-
    >-
  ]
  <<<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-
    >
  +]-
]
```

**Tape**

```
... 0 0 0 0 0 0 0 0 2 255 104 ...
```

**Input**

```
ello
```

**Output**

```
01101000
01100101
01101100
01101111
```
Example: ASCII to binary

```
- 
[ 
  >, 
  [ 
    << 
    [-<] 
    + 
    +[->]+- 
    >- 
  ] 
<<<<<<< 
+[- 
  <+++++++ 
  [->+++++[>]. 
  [-] 
  > 
  +] - 
]
```

Tape

```
... 0 0 0 0 0 0 0 1 255 104 ...
```

Input

```
ello
```

Output

```
01101000 
01100101 01101100 01101111
```
Example: ASCII to binary

```
- [ >, ] <<<[+->]+->>>-

Tape
... 0 0 0 0 0 0 0 1 255 104 ...

Input
ello

Output
```

Tape
... 0 0 0 0 0 0 0 1 255 104 ...

Input
ello

Output
Example: ASCII to binary

```
-  
[  
  >,  
  [  
    <<  
    [-<]  
    +  
    [+->]+-  
    >-  
  ]  
<<<<<<<<<
+[-  
  <+++++++  
  [->++++++<]> .  
  [-]  
  >  
  +]-  
]
```

**Tape**

```
... 0 0 0 0 0 0 0 1 0 104 ...
```

**Input**

```
ello
```

**Output**

```
01101000
01100101011011000110110001101111
```
Example: ASCII to binary

```
- 
[ 
  >,
  [ 
    <<
    [-<] 
    + 
    +[-->]+- 
    >>-
  ]
  <<<<<<<<
+[-
  <+++++++ 
  [-++++++<]>.
  [-]
  >
  +][- 
]
```

<table>
<thead>
<tr>
<th>Tape</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>... 0 0 0 0 0 0 0 0 1 0 104 ...</td>
<td>ello</td>
<td>01101000 01100101 01101100 01101111</td>
</tr>
</tbody>
</table>
Example: ASCII to binary

```
- 
 [ 
   >, 
   [ << 
     [-<] + 
     +[->+]- >-
   ] <<<><<<<<
   +[-<<<<<<
     <+++++++ 
     [-++++++++<]> . 
     [-] > 
     +][- -
    ]
]
```

<table>
<thead>
<tr>
<th>Tape</th>
<th>... 0 0 0 0 0 0 0 0 1 255 104 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>ello</td>
</tr>
<tr>
<td>Output</td>
<td></td>
</tr>
</tbody>
</table>
Example: ASCII to binary

```

Tape
... 0 0 0 0 0 0 0 1 255 104 ...

Input
eollo

Output
```

**Input:**
`ello`

**Output:**
`01101000 01100101011011000110110001101111`
**Example: ASCII to binary**

```plaintext
- 
[
  >,
  
  [ <<
    [-<<]
    +
    +[->]+-
    >-
  ]
  <<<<<<<<
  +[-
    <+++++++[
    [->+++++[>.
    [-
    >
    +]-
  ]

Tape
...
0 0 0 0 0 0 0 0 1 255 103 ...

Input
ello

Output

```

**Output:**

```
01101000
01100101
01101100
01101111
```
Example: ASCII to binary

```
```

**Tape**

```
... 0 0 0 0 0 0 0 1 255 103 ...
```

**Input**

```
ello
```

**Output**

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
- 
[ 
  >, 
  [ 
    << 
    [->]<] 
    + 
    +[->]+- 
    >=- 
  ] 
<<<<<<<<< 
+[-
  <+++++++ 
  [->+++++[>]. 
  [-] 
  > 
  +]- 
]
```

<table>
<thead>
<tr>
<th>Tape</th>
<th>... 0 0 0 0 0 0 0 0 1 255 103 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>ello</td>
</tr>
<tr>
<td>Output</td>
<td></td>
</tr>
</tbody>
</table>
Example: ASCII to binary

```
-  
 [  
   >,  
   [  
     <<  
     [-<-]  
     +  
     +[->+]  
     >-  
   ]  
 <<<<<<  
 +[-  
   <+++++++  
   [->+++++<]>.  
   [-]  
   >  
   +]-  
 ]
```

Input
ello

Output

Tape
... 0 0 0 0 0 0 0 0 1 255 103 ...

Input
ello

Output
Example: ASCII to binary

```
- [ >, [ << [-<] + [+->]- >- ] <<<[<<<<<<+[-+++++++<]>.-[-] >+[->+++++<]>.-]]
```

Tape

```
... 0 0 0 0 0 0 0 1 255 103 ...
```

Input

```
ello
```

Output

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
- 
[ 
   >,
   [ << 
      [ -< ]
      +
      +[->]+-
   ]
] <<<<<<<<
+[- 
   <+++++++
   [->++++++]>
   [ -]
   >
   +] 
]
```

Tape

```
... 0 0 0 0 0 0 0 1 255 103 ...
```

Input

```
ello
```

Output

```
01101000
01100101
01101100
01101111
```
# Example: ASCII to binary

```
- [ 
  >, 
  [ << [-<] + +[-]>+- >-
  [-] > +][-]
]
```

<table>
<thead>
<tr>
<th>Tape</th>
<th>... 0 0 0 0 0 0 0 0 0 255 103 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>ello</td>
</tr>
<tr>
<td>Output</td>
<td>01101000 01100101 01101100 01101111</td>
</tr>
</tbody>
</table>
Example: ASCII to binary

```
- [ >, [ << [-<] + [+->]+- >> ] <<<<<<<<+ [-]<+++++++>[->++++++<]> . [-] > +][-]
```

Tape

```
... 0 0 0 0 0 0 0 0 255 103 ...
```

Input

```
ello
```

Output
Example: ASCII to binary

```
- [
  >,
  [
    <<
    [-><]
    +
    +[->]+- 
    >-
  ]
  <<<<<<<<
  +[-
    <+++++++
    [->+++++<]>
    [-]
    >
    +]-
]
```

Tape

```
... 0 0 0 0 0 0 0 0 255 103 ...
```

Input

```
hello
```

Output

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
- [ 
  >, 
  [ << [-<] + +[-]>+- >-
  ] <<<<<<<<<<< 
  +[-]<<<<+++++ 
  [-]>++++++<. 
  [-] > +] [- 
  ]
```

**Tape**

```
... 0 0 0 0 0 0 1 0 255 103 ...
```

**Input**

```
ello
```

**Output**

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
- 
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >>-
  ]
<<<<<<<<<<<
+[-
  <+++++++
  [->++++++<]>.
  [-]
  >
  +][--
]
```

Tape

... 0 0 0 0 0 0 0 2 0 255 103 ...

Input

ello

Output

... 1101000 01100101 01101100 01101111 ...
Example: ASCII to binary

```
- [ >,  
    [<[<] + +[-]>+- >- ]<<<<<<
+([-<++++++<]> [-] > +][-]
]
```

Tape

```
... 0 0 0 0 0 0 2 0 255 103 ...
```

Input

```
ello
```

Output
Example: ASCII to binary

```
-  
[  
  >,  
    [  
      <<  
        [->]  
        +  
        +[->]+  
        >  
    ]  
  ]  
<<<<<<<<<  
+[-  
  <+++++++  
    [->++++++<]>  
    [-]  
    >  
    +]-  
]
```

**Tape**

```
... 0 0 0 0 0 0 1 0 255 103 ...
```

**Input**

```
ello
```

**Output**

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
- 
[ 
  >,
  ]
  <<
  [-<]
  +
  +[->+-]
  >-
] <<<<<<<<<<<<<
  +[-
    <+++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

**Tape**

```
... 0 0 0 0 0 0 1 0 255 103 ...
```

**Input**

```
ello
```

**Output**

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
1 -
[  
  >,
    [  
      <<
        [-<>]  
        +  
        +[->+]-
        >-
    ]  
<<<<<<<<<  
+[-  
  <+++++++  
    [->+++++<]>.
    [-]  
    >  
    +]-  
]
```

Tape
```
... 0 0 0 0 0 0 0 1 1 255 103 ...
```

Input
```
ello
```

Output
```
01101000 01100101 01101100 01101111
```
**Example: ASCII to binary**

```
- 
[  
  >,  
  [  
    <<  
    [<-]  
    +  
    +[->]+-  
    >>  
  ]  
 <<<<<<<< <<<<<<<<  
+[-  
  <+++++++  
  [->+++++[>]>  
  [-  
  >  
  +]-  
]
```

**Tape**

```
... 0 0 0 0 0 0 1 1 255 103 ...
```

**Input**

```
ello
```

**Output**

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```brainfuck
- 
[ 
  >,
  [
    <<
    [<-]
    +
    +[->+]-
    ->-
  ]
<<<
  +[-
    <+++++++
    [->+++<]>.
    [-]
    >
    +]--
]
```

Tape

```
... 0 0 0 0 0 0 1 1 255 103 ...
```

Input

```
ello
```

Output

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
[ -
  [>
    [<
      [-<]
      +
      +[->]+-
      >-
    ]
  ]
<<<<<<<<<
+
<<<<+++++
[->+++++++]<.
[-]
>
+[]-
]
```

Tape
```
... 0 0 0 0 0 0 1 0 255 103 ...
```

Input
```
ello
```

Output
```
01101000
01100101
01101100
01101111
```
Example: ASCII to binary

```
- 
[ 
  >, 
  [ 
    << 
    [-<>] 
    + 
    +[->+]-- 
    >-
  ]
  <<<<<<<<<
  +[-
    <+++++++ 
    [->++++++<]>.
    [-] 
    > 
    +]-- 
]
```

Tape

```
... 0 0 0 0 0 0 1 0 255 103 ...
```

Input

```
ello
```

Output

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```brainfuck
-[
  >,
  [<<
    [->-]
    +
    +[->]+-
  ]
]
<<<<<<<<<<<
+[-
  <+++++++[
    [-]>++++++<]
  [-]
  >
  +]-
]
```

Tape

```
... 0 0 0 0 0 0 1 0 0 103 ...
```

Input

```
ello
```

Output

```
01101000
01100101
01101100
01101111
```
Example: ASCII to binary

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
- [
  >,
  [<<
  [-><]
  +
  +[->]+-
  >-

] <<<<<<<<
  +[-
  <+++++++>
  [->++++++<].
  [-]
  >
  +]-
]
```

**Tape**

```
... 0 0 0 0 0 0 1 0 0 103 ...
```

**Input**

```
ello
```

**Output**

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
- 
[ 
  >, 
  [ 
    << [-<] + [+[-]>] - 
    >= 
  ] 
<<<<<<<<< 
+[-] 
<+++++++ [-]>++++++<]. [ ] > +] - 
]
```

Tape

```
... 0 0 0 0 0 0 1 0 255 103 ...
```

Input

```
ello
```

Output

```
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```brainfuck
-  
  [ 
    >, 
    [ << 
    [-<] + [+][-]+- >-
    ]] <<<<<<<<+
    +[-]<+++< ]>. [- ] > +][-]-
  ]
Tape
... 0 0 0 0 0 0 1 0 255 103 ...
Input
ello
Output
```
Example: ASCII to binary

```
- 
[ 
 ]

[<<
 ]

[+++
 ]

[-<]

[-<]

[-+]--

[->+++<]>

[-]

[->++++++<]>

[-]

[->++++++<]>

[-]

[->++++++<]>

]]

Tape

... 0 0 0 0 0 0 1 0 255 102 ...

Input

ello

Output

01101000 01100101 01101100 01101111
Example: ASCII to binary

```plaintext
- 
[ 
  >, 
  [ << 
    [->]< 
    + 
    +[-]->- 
    >= 
  ] 
  <<<<<<<<<< 
+[- <+++++++ [->++++++++<]> . 
  [-] 
  > 
  +] - 
]

Tape
... 0 0 0 0 0 0 1 0 255 102 ...

Input
ello

Output
01101000 01100101 01101100 01101111
```
Example: ASCII to binary

```
```

Tape

```
... 0 1 1 0 1 0 0 0 255 0 ...
```

Input

dello

Output
Example: ASCII to binary

```brainfuck
- 
 [ 
  >, 
  [ << 
    [-<] + [+->+]- >>-
  ] <<<<<<<<
+[-][<++++++<[->++++++<]>.
  [-] > +]-
]
```

Tape

```
... 0 1 1 0 1 0 0 0 255 0 ...
```

Input

```
ello
```

Output
Example: ASCII to binary

```
- 
  [ 
    >, 
      [ << 
          [-<-]  
          +    
          +[->+] -  
          >>    ]  
    <<<<<<<<    
    +[-    ]  
      <+++++++    
        [->+++++[>]>.  
        [-]  
    >  
    +] - 
]  
```

**Tape**

```
... 0 0 0 0 0 0 0 0 255 0 ... 
```

**Input**

`ello`

**Output**

`01101000`
Example: ASCII to binary

```
- 
[ 
  >,
  [ <<
    [-<]
    +
    +[-]>+-
    >-
  ]
<<<<<<<<<
+[-
  <+++++++[
    [-]>+++++<].
  [<-
    >
    +]-
]
```

**Tape**

<table>
<thead>
<tr>
<th>...</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>255</th>
<th>0</th>
<th>...</th>
</tr>
</thead>
</table>

**Input**

`ello`

**Output**

`01101000`
Example: ASCII to binary

```
- [ >, 
  [ << [-<] + [+->+]- >-
  ] <<<<<<<<<
+[- <++++++ 
  [-+++++]>. 
  [-] > +]-
]
```

Tape

```
... 0 0 0 0 0 0 0 0 0 255 0 ... 
```

Input

```
ello
```

Output

```
01101000
```
Example: ASCII to binary

```
- 
[ 
  >, 
  [ 
    << 
    [<-] 
    + 
    +[-]>+- 
    >= 
  ] 
<<<<<<<<< 
+[- 
  <+>>>>>> 
  [->+>+<<]>. 
  [-+>+<]> 
  [->+>+>>]>>. 
  [>>]+[-<]<. 
] 
```

Tape

```
... 0 0 0 0 0 0 0 0 255 0 ... 
```

Input

```
hello
```

Output

```
0110100001100101011011000110110001101111
```
Significance

- Best-known esoteric programming language
Significance

- Best-known esoteric programming language
- Many implementations, like Awib
Significance

- Best-known esoteric programming language
- Many implementations, like *Awib*
- Smallest current interpreter: 98 bytes!
### Significance

- Best-known esoteric programming language
- Many implementations, like *Awib*
- Smallest current interpreter: 98 bytes!
- Someone wrote a text adventure:
Significance

- Best-known esoteric programming language
- Many implementations, like *Awib*
- Smallest current interpreter: 98 bytes!
- Someone wrote a text adventure:
Variants

At least 200 variants, including:
At least 200 variants, including:

- **DoubleFuck** (two tapes)
## Variants

At least 200 variants, including:

- **DoubleFuck** (two tapes)
- **Boolfuck** (binary cells)
Variants

At least 200 variants, including:

- **DoubleFuck** (two tapes)
- **Boolfuck** (binary cells)
- **Brainfork** (multithreading via \(Y\))
At least 200 variants, including:

- **DoubleFuck** (two tapes)
- **Boolfuck** (binary cells)
- **Brainfork** (multithreading via Y)
- **Ook!** (for orangutans)
INTERCAL

- Created in 1972 by Donald R. Woods and James M. Lyon
• Created in 1972 by Donald R. Woods and James M. Lyon
• Motivation: Be different than FORTRAN or COBOL
INTERCAL

- Created in 1972 by Donald R. Woods and James M. Lyon
- Motivation: Be different than FORTRAN or COBOL
- **Weird** names, operators and properties
INTERCAL

- Created in 1972 by Donald R. Woods and James M. Lyon
- Motivation: Be different than FORTRAN or COBOL
- **Weird** names, operators and properties
- “Compiler Language With No Pronounceable Acronym”
INTERCAL

- Created in 1972 by Donald R. Woods and James M. Lyon
- Motivation: Be different than FORTRAN or COBOL
- **Weird** names, operators and properties
- “Compiler Language With No Pronounceable Acronym”
- Fun manual!
“Under no circumstances confuse the mesh with the interleave operator, except under confusing circumstances!”
“Under no circumstances confuse the mesh with the interleave operator, except under confusing circumstances!”

“Definition of array dimensions will be discussed later in greater detail, since discussing it in less detail would be difficult.”
“Under no circumstances confuse the mesh with the interleave operator, except under confusing circumstances!"

“Definition of array dimensions will be discussed later in greater detail, since discussing it in less detail would be difficult.”

“exp represents any expression (except colloquial and facial expressions)”
“Under no circumstances confuse the mesh with the interleave operator, except under confusing circumstances!”

“Definition of array dimensions will be discussed later in greater detail, since discussing it in less detail would be difficult.”

“exp represents any expression (except colloquial and facial expressions)”

“Precedence of operators is as follows: (The remainder of this page intentionally left blank.)”
“Under no circumstances confuse the mesh with the interleave operator, except under confusing circumstances!”

“Definition of array dimensions will be discussed later in greater detail, since discussing it in less detail would be difficult.”

“exp represents any expression (except colloquial and facial expressions)”

“Precedence of operators is as follows: (The remainder of this page intentionally left blank.)”

“This footnote intentionally unreferenced.”
Example: Multiplying by two

```
1
2
3
DO COME FROM (42)
4
5
DO :1 <- .1$#0
6
DO :2 <- #65535$#1
7
DO .1 <- :1~:2
8
(42) DO READ OUT .1
9
10
PLEASE GIVE UP
```
Example: Multiplying by two

1
2
3
4
5
6
7
8
9
10
11

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1^:2

(42) DO READ OUT .1

PLEASE GIVE UP
```
Example: Multiplying by two

Input

FIVE FOUR

.1 = 110110

PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
Example: Multiplying by two

```plaintext
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1#$0
DO :2 <- #65535#$1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

FIVE FOUR

.1 = 110110
Example: Multiplying by two

PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP

Input
FIVE FOUR

.1 = 110110
Example: Multiplying by two

```
1 PLEASE WRITE IN .1
2 DO COME FROM (42)
3 DO :1 <- .1$#0
4 DO :2 <- #65535$#1
5 DO .1 <- :1~:2
6 (42) DO READ OUT .1
7 PLEASE GIVE UP
```

**Input**

FIVE FOUR

.1 = 110110
:1 = 110110 $ 000000

CVIII
CCXVI
CDXXXII

...
Example: Multiplying by two

1
PLEASE WRITE IN .1

2
DO COME FROM (42)

3
DO :1 <- .1$#0

4
DO :2 <- #65535 $#1

5
DO .1 <- :1~:2

6
(42) DO READ OUT .1

7

8
PLEASE GIVE UP

9

10

11

Input

FIVE FOUR

.1 = 110110

:1 = 101000101000
Example: Multiplying by two

```
PLEASE WRITE IN .1
DO COME FROM (42)
DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2
(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

FIVE FOUR

.1 = 110110
:1 = 101000101000
Example: Multiplying by two

```
1 PLEASE WRITE IN .1
2 DO COME FROM (42)
3 DO :1 <- .1$#0
4 DO :2 <- #65535$#1
5 DO .1 <- :1~:2
6 (42) DO READ OUT .1
7
8 PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 110110
:1 = 101000101000
:2 = ...111111 $ ...000001
```

Output

```
CVIII
CCXVI
CDXXXII
... 
ICL275I: DON'T BYTE OFF MORE THAN YOU CAN CHEW
```
Example: Multiplying by two

1
2
3
4
5
6
7
8
9
10
11

PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0

DO :2 <- #65535$#1

DO .1 <- :1˜:2

(42) DO READ OUT .1

PLEASE GIVE UP

Input

FIVE FOUR

.1 = 110110

:1 = 101000101000

:2 = 101010101011
Example: Multiplying by two

```
PLEASE WRITE IN .1
DO COME FROM (42)
DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2
(42) DO READ OUT .1
PLEASE GIVE UP
```

Input
FIVE FOUR

.1 = 110110
:1 = 101000101000
:2 = 101010101011
Example: Multiplying by two

PLEASE WRITE IN .1
DO COME FROM (42)
DO :1 <- .1#$0
DO :2 <- #65535#$1
DO .1 <- :1~:2
(42) DO READ OUT .1
PLEASE GIVE UP

Input
FIVE FOUR

.1 = 1 1 0 1 1 0 0
:1 = 101000101000
:2 = 101010101011
Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0

DO :2 <- #65535$#1

DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

FIVE FOUR

.1 = 1 1 0 1 1 0 0

:1 = 101000101000

:2 = 101010101011
Example: Multiplying by two

```plaintext
1
PLEASE WRITE IN .1
2
DO COME FROM (42)
3
DO :1 <- .1$#0
4
DO :2 <- #65535$#1
5
DO .1 <- :1~:2
6
(42) DO READ OUT .1
7
PLEASE GIVE UP
```

Input
FIVE FOUR

.1 = 1 1 0 1 1 00
:j = 101000101000
:2 = 101010101011

Output
CVIII
Example: Multiplying by two

```
PLEASE WRITE IN .1
2
DO COME FROM (42)
3
DO :1 <- .1$#0
4
DO :2 <- #65535$#1
5
DO .1 <- :1~:2
6
(42) DO READ OUT .1
7
PLEASE GIVE UP
```

Input
FIVE FOUR

```
.1 = 1 1 0 1 1 00
:1 = 101000101000
:2 = 101010101011
```

Output
CVIII
CCXVI
CDXXXII
...
ICL275I: DON'T BYTE OFF MORE THAN YOU CAN CHEW
Example: Multiplying by two

```
1
PLEASE WRITE IN .1
2
DO COME FROM (42)
3
DO :1 <- .1$#0
4
DO :2 <- #65535$#1
5
DO .1 <- :1˜:2
6
(42) DO READ OUT .1
7
PLEASE GIVE UP

Input
FIVE FOUR

.1 = 1 1 0 1 1 0
:1 = 101000101000
:2 = 101010101011

Output
CVIII
CCXVI
CDXXXII
...
ICL275I: DON’T BYTE OFF MORE THAN YOU CAN CHEW
```
Significance

- Eric Raymond released **C-INTERCAL** in 1990
Significance

- Eric Raymond released **C-INTERCAL** in 1990
- “Large”, active community
Significance

- Eric Raymond released **C-INTERCAL** in 1990
- “Large”, active community
- Donald Knuth wrote a bug report in 2010
Significance

- Eric Raymond released **C-INTERCAL** in 1990
- “Large”, active community
- Donald Knuth wrote a bug report in 2010
- Google released a style guide!
Significance

- Eric Raymond released **C-INTERCAL** in 1990
- “Large”, active community
- Donald Knuth wrote a bug report in 2010
- Google released a style guide!

Here is an illustrative example.

**Bad:**

```
DO :3 <- "'yyyy'.1$:'1~#32768'~"#1109$#1'":1~#128'~#2735'$':1~"#
#546$#0'":~"#43679'":1~"#1365$#0'":~"#1023$#63'":'":'.1$#0
"~#34959'$':1~"#0$#1170'":~#11007'$':1~"#0$#2925'":'":"#2005$#255''
```

**Good:**

```
DO :3 <- "'yyyy'.1$:'1~#32768'~"#1109$#1'":1~#128'~#2735'$':1~"#
#546$#0'":~"#43679'":1~"#1365$#0'":~"#1023$#63'":'":'.1$#0~
#34959'$':1~"#0$#1170'":~#11007'$':1~"#0$#2925'":'":"#2005$#255''
```
## Variants

- **TriINTERCAL** (operates on ternary values)
Variants

- **TriINTERCAL** (operates on ternary values)
- **Threaded INTERCAL** (handles COME FROM statements referencing the same line)
Variants

- **TriINTERCAL** (operates on ternary values)
- **Threaded INTERCAL** (handles COME FROM statements referencing the same line)
- **Backtracking INTERCAL** (introduces the MAYBE label)
**Befunge**

- Created in 1993 by Chris Pressey
Befunge

- Created in 1993 by Chris Pressey
- Motivation: be difficult to parse
## Befunge

- Created in 1993 by Chris Pressey
- Motivation: be difficult to parse
- First **two-dimensional** language
Befunge

- Created in 1993 by Chris Pressey
- Motivation: be difficult to parse
- First **two-dimensional** language
- “Befunge” mistyping of “before”
Examples

1
> v

2
^ <
Examples

1. `>`
2. `^<`
Examples

1
> v

2
^ <
Examples

1

\[ >v \]

2

\[ ^< \]
Examples

1
> v

2
^ <
Examples

1

\[ \uparrow v \]

\[ ^< \]

1

\[ v > 0 \]

\[ v \]

2

\[ > ? < . < \]

3

\[ > 1 \]

\[ ^ \]

Output:

1011110010

Output:

42
Examples

1 >v
2 ^<

1 v>0 v
2 >?><.<
3 >1 ^
Examples

1. `>v`
2. `^<`

1. `v>0 v`
2. `?>.<.<`
3. `>1 ^`
# Examples

<table>
<thead>
<tr>
<th></th>
<th>Brainfuck</th>
<th>INTERCAL</th>
<th>Befunge</th>
<th>Malbolge</th>
<th>Shakespeare</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&amp;v</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>^&lt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>v&gt;0 v</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>&gt;?&lt;.&lt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>&gt;1 ^</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Output:**

```
1011110010
```

```
42
```
Examples

1
> v

2
^ <

1
v > 0 v

2
> ? < . <

3
> 1 ^
Examples

1
> v

1
^ <

1
v > 0 v
2
> ? < . <
3
> 1 ^
Examples

```
1  >v
2   ^<
```

```
1  v>0  v
2  >?<.<
3   >1  ^
```

Output: 1011110010
Output: 42
Examples

1. `>v`
2. `^<`

Output:
```
1011110010
```

1. `v>0 v`
2. `?>.<.<`
3. `>1 ^`

Output:
```
42
```
### Examples

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>&gt;v</code></td>
</tr>
<tr>
<td>2</td>
<td><code>^&lt;</code></td>
</tr>
<tr>
<td>3</td>
<td><code>v&gt;0 v</code></td>
</tr>
<tr>
<td>2</td>
<td><code>?&gt;&lt;.&gt;</code></td>
</tr>
<tr>
<td>3</td>
<td><code>&gt;1 ^</code></td>
</tr>
</tbody>
</table>

Output: `1011110010`...

```
666*+.@
```

Output: `42`
Examples

1  >v
2   ^<

1  v>0  v
2   >?<..<
3   >1  ^

Output:
1011110010
...
666*+.@
Output: 42
Examples

1  \texttt{>v}
2  \texttt{^<}

1  \texttt{v>0 v}
2  \texttt{>?<.<}
3  \texttt{>1 ^}

Output: 1011110010
Output: 42
Examples

1
>\nu
2
^<

1
v>0 \ v
2
>?<<
3
>1 ^

Output: 1011110010...
Examples

![Example Code](image)

Output: `1011110010...`

![Example Code](image)
Examples

1. `>v`
2. `^<`

Output: 101110010...

1. `v>0 v`
2. `>??..<`
3. `>1 ^`

1. `666++.@`
### Examples

<table>
<thead>
<tr>
<th></th>
<th>Brainfuck</th>
<th>INTERCAL</th>
<th>Befunge</th>
<th>Malbolge</th>
<th>Shakespeare</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. `>v  
   ^<`

1. `v>0  v  
   >?<.<  
   >1  ^`

1. `666*+.@`

Output: 1011110010...
Examples

1
2

>\n
^<

1
2
3

v>0 v
>?>.<.
>1 ^

Output: 1011110010...

1

666*+.@
Examples

1  >v
2   ^<

Output: 1011110010...

1  v>0  v
2   >?<<.<
3       >1  ^

1  666*+.@
Examples

```
1  >v
2   ~<

1  v>0  v
2  >?<.<
3   >1  ~

1  666*+.@
```

Output: 1011110010...
Examples

```
1  >v
2  ^<

1  v>0  v
2  ?>.<.<
3  >1  ^
```

Output: 1011110010...

```
1  666*+.@
```
Examples

1. `>v`
2. `^<`

Output:

```
1011110010...
```

1. `v>0 v`
2. `?>.<.`
3. `>1 ^`

Output: 42

1. `666*+.@`
Examples

1. `>v`  
2. `^<`

Output: 101110010...

1. `v>0 v`  
2. `?>.<.<`  
3. `>1 ^`

Output: 42

1. `666*+.@`

Output: 42

1. `>`v  
2. `^<`

1. `v>0 v`  
2. `?>.<.<`  
3. `>1 ^`

Output: 101110010...

1. `666*+.@`

Output: 42
Example: ROT13

Stack (bottom → top)

(empty)

Input
hello, world!

Output
**Example: ROT13**

```plaintext
> ~: "```!v
,v _:"z"`v
^ _:"m"`v
^ -4-9<
^ +4+9<
^ -:"M"`
^ -:"Z"`
>:"@"`!`
```

**Stack (bottom → top)**

(.empty)

**Input**

hello, world!

**Output**
Example: ROT13

```
> : "' " !
  , v  _ : "z' v
  ^  _ : "m' v
  ^ -4-9<
  | <
  ^ +4+9<
  ^ _ : "M' ^
  ^ _ : "Z' ^
  > : "@' !
```

Stack (bottom → top)

104

Input

ello, world!

Output

uoryyb, jbeyq!
Example: ROT13

```
>~:""""!v
,v _:"z"v
^ _:"m"v
^ -4-9<
| <
^ +4+9<
^ _:"M"v
^ _:"Z"v
>:"@"v!
```

Stack (bottom → top)

104 104

Input
ello, world!

Output
Example: ROT13

```
>~: ""'' !v
, v _: "z" 'v
^ _: "m" 'v
^ -4-9<
| <
^ +4+9<
^ _: "M" 'v
^ _: "Z" 'v
>: "@" 'v
```

Stack (bottom → top)

104 104

Input

ello, world!

Output

uryyb, jbeyq!
Example: ROT13

```
>~:"'"'!v
 ,v _:"z"v
 ^ _:"m"v
 ^ -4-9<
 | <
 ^ +4+9<
 ^ _:"M"v
 ^ _:"Z"v
 >:"@"!^
```

Stack (bottom → top)

```
104 104 96
```

Input

dello, world!

Output
Example: ROT13

```
1 >":"":"":"!v
2 ,v _:"z""v
3 ^ _:"m""v
4 ^ -4-9<
5 | <
6 ^ +4+9<
7 ^ _:"M""v
8 ^ _:"Z""v
9 >:""":"!v
```

**Stack (bottom → top)**

```
104 104 96
```

**Input**

```
ello, world!
```

**Output**

```
uvyyb, jbeyq!
```
Example: ROT13

```
>~:""'!v
,v_:"z"v
^_:"m"v
^ _:"M"v
^ +4+9<
^ _:"Z"v
-_:"M"v
>:"@"!~
```

**Stack (bottom → top)**

104 1

**Input**

ello, world!

**Output**

uryyb, jbeyq!
Example: ROT13

Stack (bottom → top)

104 0

Input
ello, world!

Output
Example: ROT13

```
>~: "'"'!'v
,v _:"z"'v
^ _:"m"'v
^ -4-9<
| <
^ +4+9<
^ _:"M"'v
^ _:"Z"'v
>: "@"'!~
```

Stack (bottom → top)

104 0

Input

ello, world!

Output

uryyb, jbeyq!
Example: ROT13

Stack (bottom → top)
104

Input
ello, world!

Output
Example: ROT13

Stack (bottom → top)
104 104

Input
ello, world!

Output
uryyb, jbeyq!
Example: ROT13

```
1 >~":""""!'v
2 ,v"~_:"z"'v
3 ^ "~_:"m"'v
4 ^ -4-9<
5 | <
6 ^ +4+9<
7 ^ "~_:"M"'v
8 ^ "~_:"Z"'v
9 >:"@"!'v
```

Stack (bottom → top)
104 104

Input
ello, world!

Output
u ryyb, jbeyq!
Example: ROT13

Stack (bottom → top)
104 104 122

Input
ello, world!

Output
Example: ROT13

Stack (bottom → top)
104 104 122

Input
eollo, world!

Output
uoryyb, jbeyq!
Example: ROT13

```
>~:""""!v
,\_":"z"'v
^\_":"m"'v
^-4-9<
| <
^+4+9<
^":"M"'\n^":"Z"'\n>:"@"'!^    
```

<table>
<thead>
<tr>
<th>Stack (bottom → top)</th>
</tr>
</thead>
<tbody>
<tr>
<td>104 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>ello, world!</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
</tr>
</thead>
</table>

Example: ROT13

```
>~:""""!v
,v   _:"z"'v
^     _:"m"'v
^   -4-9<
    |    <
^   +4+9<
^                        
^     _:"M"'v
^     _:"Z"'v
>:"@"'!v
```

Stack (bottom → top)
104 0

Input
ello, world!

Output
Example: ROT13

Stack (bottom → top)

104

Input

ello, world!

Output

uryyb, jbeyq!
Example: ROT13

```
>~"'"'!'v
,v    _:"z"'v
^  
^  
^  
^  
^  
^  
>:"@"!'^  
```

Stack (bottom → top)

```
104 104
```

Input

```
ello, world!
```

Output
Example: ROT13

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&gt;: &quot; &quot;!</td>
<td>v</td>
<td>v</td>
<td>_ : &quot;z&quot; v</td>
<td>_ : &quot;m&quot; v</td>
<td>^ -4-9&lt;</td>
<td>&lt;</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>,v</td>
<td>_ : &quot;M&quot; ^</td>
<td>^ +4+9&lt;</td>
<td>^ _ : &quot;Z&quot; ^</td>
<td>^ _ : &quot;@&quot; !</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>~ ^</td>
<td>^ ^ ^ ^ ^</td>
<td>^</td>
<td>^</td>
<td>^</td>
<td>^</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stack (bottom → top)

104 104

Input
dello, world!

Output

u ryyb, jbeyq!
Example: ROT13

Stack (bottom \(\rightarrow\) top)

```
104 104 109
```

Input

ello, world!

Output

uoryyb, jbeyq!
Example: ROT13

Stack (bottom → top)
104 104 109

Input
eollo, world!

Output
Example: ROT13

```
1>":",":\'!v
2   v_:"z"\'v
3   ^_:"m"\'v
4   ^-4-9<
5       |<
6   ^+4+9<
7   ^_:"M"\'<
8   ^_:"Z"\'<
9 >:"@"\'!^  
```

Stack (bottom → top)

104 0

Input  
ello, world!

Output
Example: ROT13

Stack (bottom → top)

104 0

Input
ello, world!

Output
uryyb, jbeyq!
Example: ROT13

```
>~`;""""!v
,v _:"z"v
^ _:"m"v
^ -4-9<
| <
^ +4+9<
^ _:"M"^ 
^ _:"Z"^ 
>:"@"!`
```

Stack (bottom → top)
104 0

Input
ello, world!

Output
Example: ROT13

```
>~:""""!v
,v=_:"Z"'v
^_="m"'v
^ -4-9<
| <
^ +4+9<
^_="M"'v
^_="Z"'v
>:"@"!'^
```

Stack (bottom → top)

104

Input

dello, world!

Output
Example: ROT13

```
>~:"^"^!v
, v _:"z"^v
^ _:"m"^v
^-4-9<
| <
^ +4+9<
^ "M"^v
^ "Z"^v
>:"@"^!~
```
Example: ROT13

Stack (bottom → top)
104 9

Input
ello, world!

Output
Example: ROT13

```
>~":""!v
',v"_":"z"!v
^"_<
^-4-9<
|<
^-+4+9<
^"_<
^-":"M"`
^-"Z"`
>:"@"!`
```

Stack (bottom → top)

```
113
```

Input

```
ello, world!
```

Output
Example: ROT13

Stack (bottom → top)
113 4

Input
eollo, world!

Output
Example: ROT13

```
>~":'"''@v
,v_:"z"'v
^_:"m"'v
^-4-9<
|<
^+4+9<
^_:"M''v
^_:"Z''v
>:"@"'!~
```

Stack (bottom → top)

```
117
```

Input

```
ello, world!
```

Output
Example: ROT13

```
1>~:""":!:v
2,.v_::"z":v
3^-_::"m":v
4^-`-4-9<
5|<
6^-`+4+9<
7^-`":M":`
8^-`":Z":`
9>:"@":!
```

Stack (bottom → top)

117

Input

ello, world!

Output

uryyb, jbeyq!
Example: ROT13

Stack (bottom → top)

117

Input

ello, world!

Output
Example: ROT13

```
1>":"v'
v
, v _ : "z" v
v
^ _ : "m" v
^  -4-9<
| <
^  +4+9<
^ _ : "M" ^
^ _ : "Z" ^
> : "@" ! ^
```

Stack (bottom → top)
117

Input
ello, world!

Output
Example: ROT13

```
>~""""!v
, v  _: "z" ' v
^  _: "m" ' v
^ -4-9<
| <
^ +4+9<
^  _: "M" ^
^  _: "Z" ^
> : "@" ! ~
```

Stack (bottom → top)

(empty)

Input
ello, world!

Output
u
Example: ROT13

Stack (bottom → top)
(empty)

Input

Output
uryyb, jbeyq!
Significance

- Important platform: *Befunge Mailing List*
Significance

- Important platform: _Befunge Mailing List_
- Many actively maintained interpreters and compilers, like _befunjit_
Significance

- Important platform: *Befunge Mailing List*
- Many actively maintained interpreters and compilers, like *befunjit*
- IRC client with 10,000 characters
Variants

- Other members of the *Funge-98* family: *Unefunge* and *Trefunge*
Variants

- Other members of the *Funge-98* family: **Unefunge** and **Trefunge**
- **Weird** (only one instruction)
Variants

- Other members of the *Funge-98* family: **Unefunge** and **Trefunge**
- **Weird** (only one instruction)
- **PATH** (1D source code, 2D playfield)
Malbolge

- Created in 1998 by Ben Olmstead
Malbolge

- Created in 1998 by Ben Olmstead
- Motivation: be incomprehensible and hard to use
Malbolge

- Created in 1998 by Ben Olmstead
- Motivation: be incomprehensible and **hard** to use
- Took two years to write the first nontrivial program
Malbolge

- Created in 1998 by Ben Olmstead
- Motivation: be incomprehensible and **hard** to use
- Took two years to write the first nontrivial program
- *Malebolge* is the eighth circle of Hell in Dante’s *Inferno*
Description

- Simple virtual machine
Description

- Simple virtual machine
- CPU with three registers $A$, $C$, and $D$
Description

- Simple virtual machine
- CPU with three registers $A$, $C$, and $D$
- $3^{10}$ memory cells, 10 trits each
Description

- Simple virtual machine
- CPU with three registers $A$, $C$, and $D$
- $3^{10}$ memory cells, 10 trits each

Execution

For each instruction:
Description

- Simple virtual machine
- CPU with three registers $A$, $C$, and $D$
- $3^{10}$ memory cells, 10 trits each

Execution

For each instruction:

- Subtract 33, add $C$, mod with 94
Description

- Simple virtual machine
- CPU with three registers $A$, $C$, and $D$
- $3^{10}$ memory cells, 10 trits each

Execution

For each instruction:
- Subtract 33, add $C$, mod with 94
- Apply a substitution encryption
Description

- Simple virtual machine
- CPU with three registers $A$, $C$, and $D$
- $3^{10}$ memory cells, 10 trits each

Execution

For each instruction:
- Subtract 33, add $C$, mod with 94
- Apply a substitution encryption
- If we now have one of $j \ i \ * \ p \ / \ < \ v \ o$, execute that instruction
Description

- Simple virtual machine
- CPU with three registers $A$, $C$, and $D$
- $3^{10}$ memory cells, 10 trits each

Execution

For each instruction:
- Subtract 33, add $C$, mod with 94
- Apply a substitution encryption
- If we now have one of $j ~ i * p / < v ~ o$, execute that instruction
- Subtract 33
Description

- Simple virtual machine
- CPU with three registers $A$, $C$, and $D$
- $3^{10}$ memory cells, 10 trits each

Execution

For each instruction:
- Subtract 33, add $C$, mod with 94
- Apply a substitution encryption
- If we now have one of $j\ i\ *\ p\ /\ <\ v\ o$, execute that instruction
- Subtract 33
- Apply a different substitution encryption
Description

- Simple virtual machine
- CPU with three registers $A$, $C$, and $D$
- $3^{10}$ memory cells, 10 trits each

Execution

For each instruction:
- Subtract 33, add $C$, mod with 94
- Apply a substitution encryption
- If we now have one of $j \ i \ * \ p \ / \ < \ v \ o$, execute that instruction
- Subtract 33
- Apply a different substitution encryption
- Increment $C$ and $D$
Example: Hello world

```
1 (=<``$9]7<5YXz7wT.3,+0/o'K%$H"~D|#z@b=`{~Lx8%$X
2 mrkpohm-kNi;gsedcba`_`\[ZYXWVUTSRQPONMLKJIHGFE
3 DCBA@?>=<.;9876543s+O<oLm
```
Example: Hello world

```
1 (=<`$9]7<5YXz7wT.3,+0/o'K%$H" '~D|#z@b=`{~$8%$X
2 mrkpohm-kNi;gsedcba`_`)\[ZYXWVUTSRQPONMLKJHGFE
3 DCBA>?>=<;:9876543s+0<oLm
```

Output

```
HELL0 WOR1d
```
Significance

- Cryptanalysis by Louis Scheffer around 2005
Significance

- Cryptanalysis by Louis Scheffer around 2005
- Appearance in *Elementary* S01E10:
Shakespeare

- Created in 2001 by Karl Hasselström and Jon Åslund
Shakespeare

- Created in 2001 by Karl Hasselström and Jon Åslund
- Motivation: homework in their Syntax Analysis class
Shakespeare

- Created in 2001 by Karl Hasselström and Jon Åslund
- Motivation: homework in their Syntax Analysis class
- **Themed** language
Shakespeare

- Created in 2001 by Karl Hasselström and Jon Åslund
- Motivation: homework in their Syntax Analysis class
- Themed language
- “combines the expressiveness of BASIC with the user-friendliness of assembly language”
Example: Fibonacci sequence

A drama by the numbers.

Juliet, a young Italian lady.
Romeo, the rich Count.
Mercutio, his spacy rival.

Variables

Output
Example: Fibonacci sequence

1. A drama by the numbers.
2. Juliet, a young Italian lady.
3. Romeo, the rich Count.
4. Mercutio, his spacy rival.

Variables

Output
Example: Fibonacci sequence

A drama by the numbers.

Juliet, a young Italian lady.
Romeo, the rich Count.
Mercutio, his spacy rival.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Juliet</th>
<th>0</th>
<th>off</th>
</tr>
</thead>
</table>

Output
Example: Fibonacci sequence

1. A drama by the numbers.
2. Juliet, a young Italian lady.
3. Romeo, the rich Count.
4. Mercutio, his spacy rival.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Juliet</th>
<th>Romeo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>off</td>
<td>off</td>
</tr>
</tbody>
</table>

Output
Example: Fibonacci sequence

A drama by the numbers.

Juliet, a young Italian lady.
Romeo, the rich Count.
Mercutio, his spacy rival.

Variables

<table>
<thead>
<tr>
<th></th>
<th>Juliet</th>
<th>Romeo</th>
<th>Mercutio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>State</td>
<td>off</td>
<td>off</td>
<td>off</td>
</tr>
</tbody>
</table>

Output
Example: Fibonacci sequence

---

**Act I: The Act where it all happens.**

Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!

Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]

[Enter Romeo]

Juliet: You devil! You are nothing!

Romeo: Open your heart! Remember me!

[Exit Juliet]
Example: Fibonacci sequence

Act I: The Act where it all happens.

   Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]

[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
Example: Fibonacci sequence

Act I: The Act where it all happens.
Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
Example: Fibonacci sequence

Act I: The Act where it all happens.
Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
Example: Fibonacci sequence

Act I: The Act where it all happens.
   Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]

[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]

Variables

<table>
<thead>
<tr>
<th></th>
<th>Juliet</th>
<th>0</th>
<th>on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>0</td>
<td></td>
<td>off</td>
</tr>
<tr>
<td>Mercutio</td>
<td>0</td>
<td></td>
<td>on</td>
</tr>
</tbody>
</table>
Example: Fibonacci sequence

Act I: The Act where it all happens.
Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]

[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]

Variables

<table>
<thead>
<tr>
<th></th>
<th>Juliet</th>
<th>Romeo</th>
<th>Mercutio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1</td>
<td>0</td>
<td>-32</td>
</tr>
</tbody>
</table>

Output

1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
28657
46368
75025
121393
196418
317811
514229
832040
1346269
2178309
3524578
Example: Fibonacci sequence

Act I: The Act where it all happens.
Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
Example: Fibonacci sequence

Act I: The Act where it all happens.
Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]

[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
Example: Fibonacci sequence

Act I: The Act where it all happens.
   Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]

[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
Example: Fibonacci sequence

Act I: The Act where it all happens.
Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
Example: Fibonacci sequence

Act I: The Act where it all happens.
Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
Example: Fibonacci sequence

Act I: The Act where it all happens.
Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
Example: Fibonacci sequence

```
Act I: The Act where it all happens.
   Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Juliet</td>
<td>1 (0)</td>
<td>on</td>
</tr>
<tr>
<td>Romeo</td>
<td>0</td>
<td>on</td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>off</td>
</tr>
</tbody>
</table>

Output

1
Example: Fibonacci sequence

Act I: The Act where it all happens.
   Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as beautiful as a flower!
Juliet: You are a disgusting smelly lying rotten dirty pig! You are as small as the difference between nothing and thyself!

[Exit Mercutio]

[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]

Variables

<table>
<thead>
<tr>
<th>Juliet</th>
<th>1 (0)</th>
<th>off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>0</td>
<td>on</td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>off</td>
</tr>
</tbody>
</table>

Output

1

2 3 5 8 13 21 34 55 89 144
233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578
Example: Fibonacci sequence

```
Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

[Exit Mercutio]
```

Variables

<table>
<thead>
<tr>
<th></th>
<th>Juliet</th>
<th>Romeo</th>
<th>Mercutio</th>
</tr>
</thead>
<tbody>
<tr>
<td>on/off</td>
<td>1 (0)</td>
<td>0</td>
<td>32</td>
</tr>
</tbody>
</table>

Output

1
Example: Fibonacci sequence

26  Scene II: The rival’s encounter.

28  [Enter Mercutio]

29  Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

32  Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

37  [Exit Mercutio]

Variables

<table>
<thead>
<tr>
<th></th>
<th>Juliet</th>
<th>Romeo</th>
<th>Mercutio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juliet</td>
<td>1 (0)</td>
<td>off</td>
<td></td>
</tr>
<tr>
<td>Romeo</td>
<td>0</td>
<td>on</td>
<td></td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>on</td>
<td></td>
</tr>
</tbody>
</table>

Output

1
Example: Fibonacci sequence

Scene II: The rival’s encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

[Exit Mercutio]

Variables

<table>
<thead>
<tr>
<th></th>
<th>Juliet</th>
<th>Romeo</th>
<th>Mercutio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 (0)</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>off</td>
<td>on</td>
<td>on</td>
</tr>
</tbody>
</table>

Output

1
Example: Fibonacci sequence

Scene II: The rival’s encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

[Exit Mercutio]

<table>
<thead>
<tr>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juliet</td>
</tr>
<tr>
<td>Romeo</td>
</tr>
<tr>
<td>Mercutio</td>
</tr>
</tbody>
</table>

Output
1
Example: Fibonacci sequence

26
27
Scene II: The rival’s encounter.
28
29
[Enter Mercutio]

Romeo: Are you better than me? If not, let
us proceed to scene IV. Speak your mind!

31
32
Mercutio: You are as miserable as the sum
of thyself and a stone wall! Remember
yourself!

34
35
[Exit Mercutio]

Variables

<table>
<thead>
<tr>
<th>Juliet</th>
<th>1  (0)</th>
<th>off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>0</td>
<td>on</td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>on</td>
</tr>
</tbody>
</table>

Output

1
Example: Fibonacci sequence

```
Scene II: The rival’s encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

[Exit Mercutio]
```

**Variables**

<table>
<thead>
<tr>
<th>Juliet</th>
<th>1 (0)</th>
<th>off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>1</td>
<td>on</td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>on</td>
</tr>
</tbody>
</table>

**Output**

```
1
```
Example: Fibonacci sequence

Scene II: The rival’s encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

[Exit Mercutio]
Example: Fibonacci sequence

Scene II: The rival’s encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

[Exit Mercutio]
Example: Fibonacci sequence

Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as happy as the sum of thyself and me. Open your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!

Variables

<table>
<thead>
<tr>
<th></th>
<th>Juliet</th>
<th>Romeo</th>
<th>Mercutio</th>
</tr>
</thead>
<tbody>
<tr>
<td>on/off</td>
<td>1 (0)</td>
<td>1 (1)</td>
<td>32</td>
</tr>
</tbody>
</table>

Output

1
Example: Fibonacci sequence

Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as happy as the sum of thyself and me. Open your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!

Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juliet</td>
<td>1 (0)</td>
<td>on</td>
</tr>
<tr>
<td>Romeo</td>
<td>1 (1)</td>
<td>on</td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>off</td>
</tr>
</tbody>
</table>

Output

1
Example: Fibonacci sequence

Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as happy as the sum of thyself and me. Open your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!

Variables

<table>
<thead>
<tr>
<th></th>
<th>Juliet</th>
<th>Romeo</th>
<th>Mercutio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1 (0)</td>
<td>1 (1)</td>
<td>32</td>
</tr>
<tr>
<td>On</td>
<td>on</td>
<td>on</td>
<td>off</td>
</tr>
</tbody>
</table>

Output

\[
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040, 1346269, 2178309, 3524578
\]
Example: Fibonacci sequence

Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as happy as the sum of thyself and me. Open your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!

Variables

<table>
<thead>
<tr>
<th>Juliet</th>
<th>0</th>
<th>on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>1 (1)</td>
<td>on</td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>off</td>
</tr>
</tbody>
</table>

Output

1
Example: Fibonacci sequence

Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as happy as the sum of thyself and me. Open your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!
Example: Fibonacci sequence

Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as happy as the sum of thyself and me. Open your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!

Variables

| Juliet   | 1 | on |
| Romeo    | 1 (1) | on |
| Mercutio | 32 | off |

Output

11_1
Example: Fibonacci sequence

Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as happy as the sum of thyself and me. Open your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!

Variables

<table>
<thead>
<tr>
<th>Juliet</th>
<th>1 (1)</th>
<th>on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>1 (1)</td>
<td>on</td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>off</td>
</tr>
</tbody>
</table>

Output

\(1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040, 1346269, 2178309, 3524578\)
Example: Fibonacci sequence

Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as happy as the sum of thyself and me. Open your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!

---

**Variables**

<table>
<thead>
<tr>
<th>Juliet</th>
<th>1 (1)</th>
<th>on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>1</td>
<td>on</td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>off</td>
</tr>
</tbody>
</table>

**Output**

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040, 1346269, 2178309, 3524578
Example: Fibonacci sequence

Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as happy as the sum of thyself and me. Open your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!

Variables

<table>
<thead>
<tr>
<th>Juliet</th>
<th>1 (1)</th>
<th>off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>1</td>
<td>on</td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>off</td>
</tr>
</tbody>
</table>

Output

1

1

1

1

2

3

5

8

13

21

34

55

89

144

233

377

610

987

1597

2584

4181

6765

10946

17711

28657

46368

75025

121393

196418

317811

514229

832040

1346269

2178309

3524578
Example: Fibonacci sequence

Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as happy as the sum of thyself and me. Open your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!

### Variables

<table>
<thead>
<tr>
<th></th>
<th>Juliet</th>
<th>Romeo</th>
<th>Mercutio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1 (1)</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>State</td>
<td>off</td>
<td>on</td>
<td>off</td>
</tr>
</tbody>
</table>

### Output

$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040, 1346269, 2178309, 3524578$
Example: Fibonacci sequence

Scene II: The rival’s encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

[Exit Mercutio]

Variables

<table>
<thead>
<tr>
<th>Juliet</th>
<th>1 (1)</th>
<th>off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>1</td>
<td>on</td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>off</td>
</tr>
</tbody>
</table>

Output

1, 1
**Example: Fibonacci sequence**

```plaintext
Scene II: The rival’s encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

[Exit Mercutio]
```

**Variables**

<table>
<thead>
<tr>
<th></th>
<th>Juliet</th>
<th>Romeo</th>
<th>Mercutio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3524578</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>off</td>
<td>on</td>
<td>off</td>
</tr>
</tbody>
</table>

**Output**

```
1 1 2 3 5 8 13 21 34 55 89 144
233 377 610 987 1597 2584 4181
6765 10946 17711 28657 46368
75025 121393 196418 317811 514229
832040 1346269 2178309 3524578
```
Example: Fibonacci sequence

Scene II: The rival’s encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

[Exit Mercutio]
Example: Fibonacci sequence

Scene II: The rival’s encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

[Exit Mercutio]

**Variables**

<table>
<thead>
<tr>
<th>Variables</th>
<th>Juliet</th>
<th>3524578</th>
<th>off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>32</td>
<td>on</td>
<td></td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>on</td>
<td></td>
</tr>
</tbody>
</table>

**Output**

1  1  2  3  5  8  13  21  34  55  89  144  
233 377 610 987 1597 2584 4181  
6765 10946 17711 28657 46368  
75025 121393 196418 317811  
514229 832040 1346269 2178309  
3524578
Example: Fibonacci sequence

Scene II: The rival’s encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum of thyself and a stone wall! Remember yourself!

[Exit Mercutio]
Example: Fibonacci sequence

Scene IV: The finale.

Mercutio: Are you better than me? You bastard.

[Exit Mercutio]
[Enter Juliet]

Romeo: You are my pretty rose!

Juliet: You coward! You are as bad as Mercutio. Recall my final goodbye.

[Exit Juliet]

Romeo: Am I as cursed as a damned hound?

Variables

<table>
<thead>
<tr>
<th>Juliet</th>
<th>3524578</th>
<th>off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>32</td>
<td>on</td>
</tr>
<tr>
<td>Mercutio</td>
<td>32</td>
<td>on</td>
</tr>
</tbody>
</table>

Output

1  1  2  3  5  8  13  21  34  55  89  144
233 377 610 987 1597 2584 4181
6765 10946 17711 28657 46368
75025 121393 196418 317811
514229 832040 1346269 2178309
3524578
Significance

- DeCSS implementation would be protected by free speech laws
Significance

- DeCSS implementation would be protected by free speech laws
- Actual Shakespeare performance in 2007:
Other themed languages

- **Chef** (recipes)
Other themed languages

- **Chef** (recipes)
- **Taxi** (directions for a taxi driver)
Other themed languages

- **Chef** *(recipes)*
- **Taxi** *(directions for a taxi driver)*
- **LOLCODE** *(IM IN YR LOOP, KTHXBYE)*
Other themed languages

- **Chef** *(recipes)*
- **Taxi** *(directions for a taxi driver)*
- **LOLCODE** *(IM IN YR LOOP, KTHXBYE)*
- **FiM++** *(Dear Princess Celestia...)*
More examples

- **Piet** (abstract paintings)
### More examples

- **Piet** (abstract paintings)
- **Aceto** (follows a Hilbert curve through the source code)
More examples

- **Piet** (abstract paintings)
- **Aceto** (follows a Hilbert curve through the source code)
- **Whitespace** (uses only space, tab, and return)
More examples

- **Piet** (abstract paintings)
- **Aceto** (follows a Hilbert curve through the source code)
- **Whitespace** (uses only space, tab, and return)
- **///** (the only operation is string substitution)
More examples

- **Piet** (abstract paintings)
- **Aceto** (follows a Hilbert curve through the source code)
- **Whitespace** (uses only space, tab, and return)
- /// (the only operation is string substitution)
- **Velato** (MIDI, tends to create jazz)
More examples

- **Piet** (abstract paintings)
- **Aceto** (follows a Hilbert curve through the source code)
- **Whitespace** (uses only space, tab, and return)
- **///** (the only operation is string substitution)
- **Velato** (MIDI, tends to create jazz)

Visit esolangs.org for (many, many) more!
Thanks!

sebastian@morr.cc
https://morr.cc
@blinry

Slides and references: morr.cc/esolangs/