1

2

3

4

5

6

8

9

10

11


STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW

12

14

15

17

18

20

-20

-16

-8

-5


STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW

-4	-3
-2	-1
0	22
27	30
33	40


STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW

42

49

66

81

99

100

-200

-144

-66

121


STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW

169

199

1/2

6.6

-1/8

-3/2

-0.1

-10.1

Pi

1024


STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW
STACK OVERFLOW

$\sqrt{2}$

-17.76

```
lambda x: 7
```

```
lambda x: 13
```

```
lambda x: 16
```

```
lambda x: 19
```

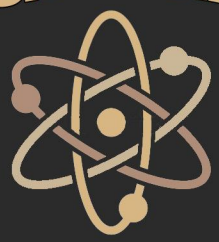
```
lambda x: x
```

```
lambda x: x + 5
```

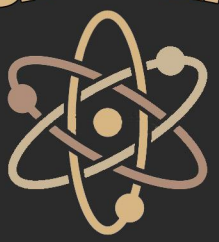
```
lambda x: x + 11
```

```
lambda x: x - 1
```

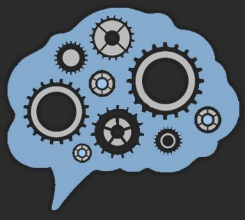
STACK OVERFLOW



STACK OVERFLOW



STACK OVERFLOW

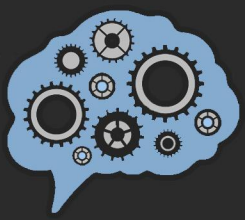


STACK OVERFLOW

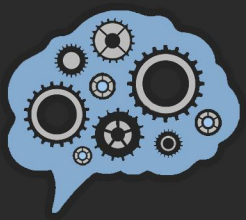
STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW



STACK OVERFLOW



STACK OVERFLOW

STACK OVERFLOW

```
lambda x: 2 * x
```

```
lambda x: min(x, 7)
```

```
lambda x: x // 3
```

```
lambda x: max(x, 13)
```

```
lambda x: gcd(x, 24)
```

```
lambda x: lcm(x, 6)
```

```
def if_prime(x):
  if x > 200:  return 0
  if is_prime(x):
    return 1

  return x
```

```
lambda x: x % 5
```

```
lambda x: x - 25
```

```
lambda x: x - 128
```

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

```
lambda x: 151 - x
```

```
lambda x: 38 - x
```

```
lambda x: x // -6
```

```
lambda x: -5 * x
```

```
lambda x: 100 // x
```

```
lambda x: x % 25
```

```
lambda x: x * (x // 4)
```

```
lambda x: ceil(sqrt(abs(x)))
```

```
lambda x: floor(log2(abs(x)))
```

```
lambda x: 3 * sin(pi * x / 2)
```

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

```python
def if_greater_or_less(x):
  if x < 5:   return 66
  if x > 66:  return 5
  return 42
```

```python
def if_equal_or_not(x):
  if   x == 10:  x += 66
  elif x != 66:  x -= 10
  else:
    return x // 10

  return x
```

```python
def for_cycle(x):
  num = floor(x) % 5
  for i in range(num):
    x += 10

  return x
```

```python
def while_cycle(x):
  while x % 2 != 0:
    x = round(x / 3)

  return x
```

```python
def int_from_list(x):
  # index:  012345678
  string = "957812463"
  return int(string[x % 9])
```

```python
def ints_from_list(x):
  # index:  0123456789
  string = "176485923074"
  a = x % 9
  b = a + (a % 3) + 1
  return int(string[a:b])
```

```python
def split_by_int(x):
  x = str(int(x) % 10)

  string = "376326492"
  parts = string.split(x)
  return len(parts) - 2
```

```python
lambda x: int(str(abs(x))[0])
```

```python
lambda x: x + 0.5
```

```python
lambda x: -0.5 * x
```

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

```python
lambda x: x / 1.5
```

```python
lambda x: 1 / x
```

```python
def switch_places(x):
  y = abs(x)
  s = "{0:.2f}".format(y)
  a, b = s.split(".")
  result = float(b + "." + a)
  return result * sign(x)
```

```python
def put_and_eval(x):
  res = "{0:.2f}".format(x)
  res.replace('.', '-')
  return eval(res)
```

```python
def reverse(x):
  string = str(abs(int(x)))
  string = string[::-1]
  res = int(string)
  return res * sign(x)
```

```python
def subtract_madness(x):
  s = "{0:.2f}".format(x)
  s = s.replace('.', '--')
  return eval('-'.join(s))
```

```python
def rec_subtract(x):
  if x <= 0:
    return x

  x -= 60
  return rec_subtract(x)
```

```python
def rec_divide(x):
  if x == 0:
    return sign(x)

  x = round(x / 3)
  return sign(x) + \
         rec_divide(x)
```

```python
def double_rec(x):
  if -1 <= x <= 10: return -x

  res = double_rec(x // 100)
  res += double_rec(x // 10)
  return res
```

```python
def rec_multiply(x):
  if x % 8 == 0:
    return x

  res = rec_multiply(2 * x)
  return res - 1
```

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW

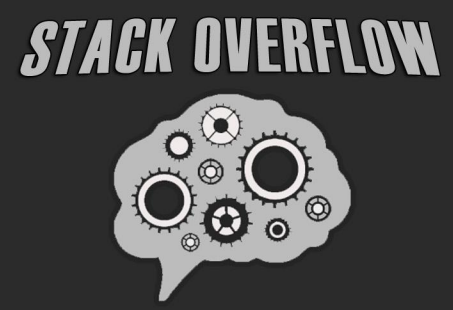
```python
def ack(m, n=None):
  if n is None:  n = m
  if m == 0:     return n + 1
  if n == 0:     n = 1
  else:
     n = ack(m, n - 1)
  return ack(m - 1, n)
```

```python
def fibb(x):
  if x <= 1: return x

  return fibb(x - 1) + \
         fibb(x - 2)
```

```python
lambda x: pow(x, -x)
```

```python
lambda x: inf
```

```python
# HOW TO USE FUNCTION:
# function: lambda x: x + 5

# input value: 3
#                ↓
#       lambda 3: 3 + 5
#                    ↓
#     output value: 8
```



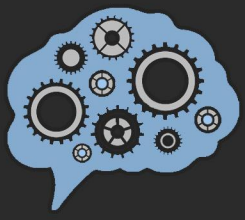

```python
# HOW SOME MYSTERIOUS THINGS WORK:

5 // 3 == 1 and -5 // 3 == -2
5 / 3 == 1.6666666666666666
5 % 3 == 2  and -5 % 3 == 1
round(0.4) == 0 and round(0.5) == 1
floor(0.7) == 0 and ceil(0.2) == 1
"asd"[1] == "s" and "asd"[0:2] == "as"
"asd"[-1] == "d" and "asd"[::-1] == "dsa"
```

```python
# HOW SOME MYSTERIOUS THINGS WORK (part 2):

def gcd(a, b):  # greatest common divisor
  while b > 0: a, b = b, a % b
  return a
def lcm(a, b): return abs(a * b) // gcd(a, b)
def is_prime(n):
  if n != int(n):  return False
  if n <= 1:       return False
  for i in range(2, n):
      if (n % i) == 0: return False
  return True
```

STACK OVERFLOW

STACK OVERFLOW

STACK OVERFLOW



STACK OVERFLOW

```python
# HOW TO PLAY THIS GAME:
def game():
  # at fist prepare the game:
  prepare_game()

  # play round as long as nobody has won
  while nobody_won():
      play_round()
```

STACK OVERFLOW

```python
# do this at the beginning of each game
def prepare_game():
  # select output value, it is the same
  # for whole game
  global output_value = pop_random_value()

  # each player gets 5 function cards
  global players = []
  for _ in range(len(players)):
      players.append(
          [pop_random_fn() for i in range(4)]
      )
```

```python
def play_round():
  input_value = pop_random_value()
  # TODO
  # each player selects some functions,
  # to transform input value to output value
  # as close as possible, the one with closest
  # value is a winner. Used functions are put
  # away, each player gets as many new functions
  # as he used, only winner gets one less.
```

```python
def nobody_won():
  # if any player has 0 functions, the game has ended
  return all(
      [len(cards) > 0 for cards in players]
  )
```