PARCIAL I - CI3641 - JOAO PINTO 17-10490

PREGUNTA 4

PARTE A I

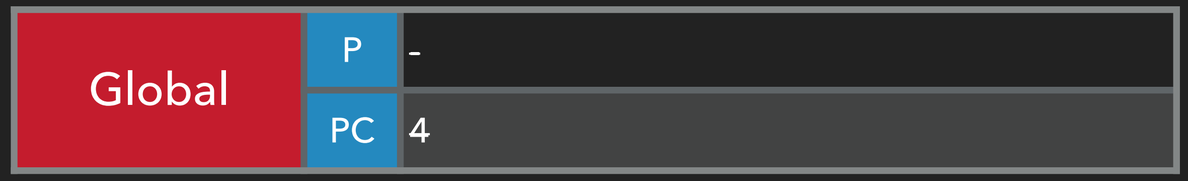
EJECUCIÓN DE ZIP

```
   def zip(a, b):
▸ 0     if a and b:
▸ 1         yield (a[0], b[0])
▸ 2         for p in zip(a[1:], b[1:]):
▸ 3             yield p

▸ 4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸ 5     print(p)
```

IMPRIME

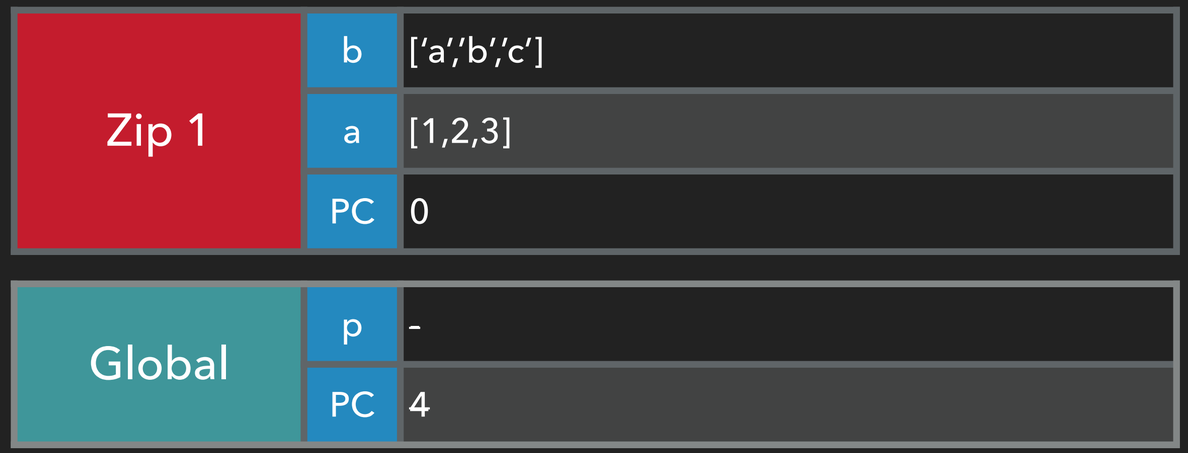
```
>
```



```
   def zip(a, b):
       if a and b:
0          yield (a[0], b[0])
1          for p in zip(a[1:], b[1:]):
2              yield p
3
4  for p in zip([1, 2, 3], ['a', 'b', 'c']):
5      print(p)
```

IMPRIME

```
>
```



```
   def zip(a, b):
▸ 0     if a and b:
▸ 1         yield (a[0], b[0])
▸ 2         for p in zip(a[1:], b[1:]):
▸ 3             yield p

▸ 4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸ 5     print(p)
```

IMPRIME

```
>
```

Zip 1	b	[‘a’,’b’,’c’]
	a	[1,2,3]
	PC	0 1
Global	p	-
	PC	4

```
   def zip(a, b):
▸0     if a and b:
▸1         yield (a[0], b[0])
▸2         for p in zip(a[1:], b[1:]):
▸3             yield p

▸4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸5     print(p)
```

IMPRIME

```
> (1,’a’)
```

Zip 1	b	[‘a’,’b’,’c’]
	a	[1,2,3]
	PC	0 1
Global	p	(1, ‘a’)
	PC	4 5

```
   def zip(a, b):
▸ 0     if a and b:
▸ 1         yield (a[0], b[0])
▸ 2         for p in zip(a[1:], b[1:]):
▸ 3             yield p

▸ 4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸ 5     print(p)
```

IMPRIME

```
> (1,'a')
```

Zip 1	p	-
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2
Global	p	(1, 'a')
	PC	4 5 4

```
   def zip(a, b):
       if a and b:
0          yield (a[0], b[0])
1          for p in zip(a[1:], b[1:]):
2              yield p
3

4  for p in zip([1, 2, 3], ['a', 'b', 'c']):
5      print(p)
```

IMPRIME

```
> (1,'a')
```

Zip 2	b	['b','c']
	a	[2,3]
	PC	0
Zip 1	p	-
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2
Global	p	(1, 'a')
	PC	4 5 4

```
   def zip(a, b):
0      if a and b:
1          yield (a[0], b[0])
2          for p in zip(a[1:], b[1:]):
3              yield p

4  for p in zip([1, 2, 3], ['a', 'b', 'c']):
5      print(p)
```

IMPRIME

```
> (1,'a')
```

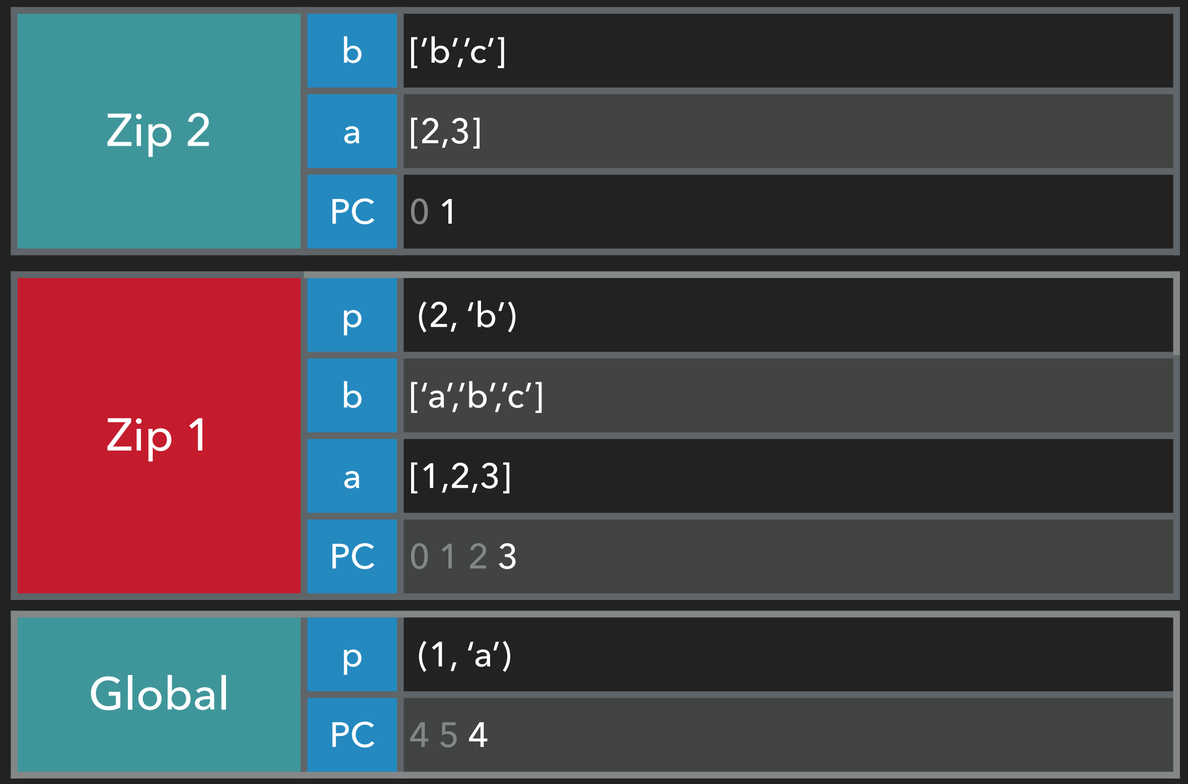
Zip 2	b	['b','c']
	a	[2,3]
	PC	0 1
Zip 1	p	-
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2
Global	p	(1, 'a')
	PC	4 5 4

```
   def zip(a, b):
▸0     if a and b:
▸1         yield (a[0], b[0])
▸2         for p in zip(a[1:], b[1:]):
▸3             yield p

▸4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸5     print(p)
```

IMPRIME

```
> (1,'a')
```



```
   def zip(a, b):
▸0     if a and b:
▸1         yield (a[0], b[0])
▸2         for p in zip(a[1:], b[1:]):
▸3             yield p

▸4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸5     print(p)
```

IMPRIME

```
> (1,’a’)
> (2,‘b’)
```

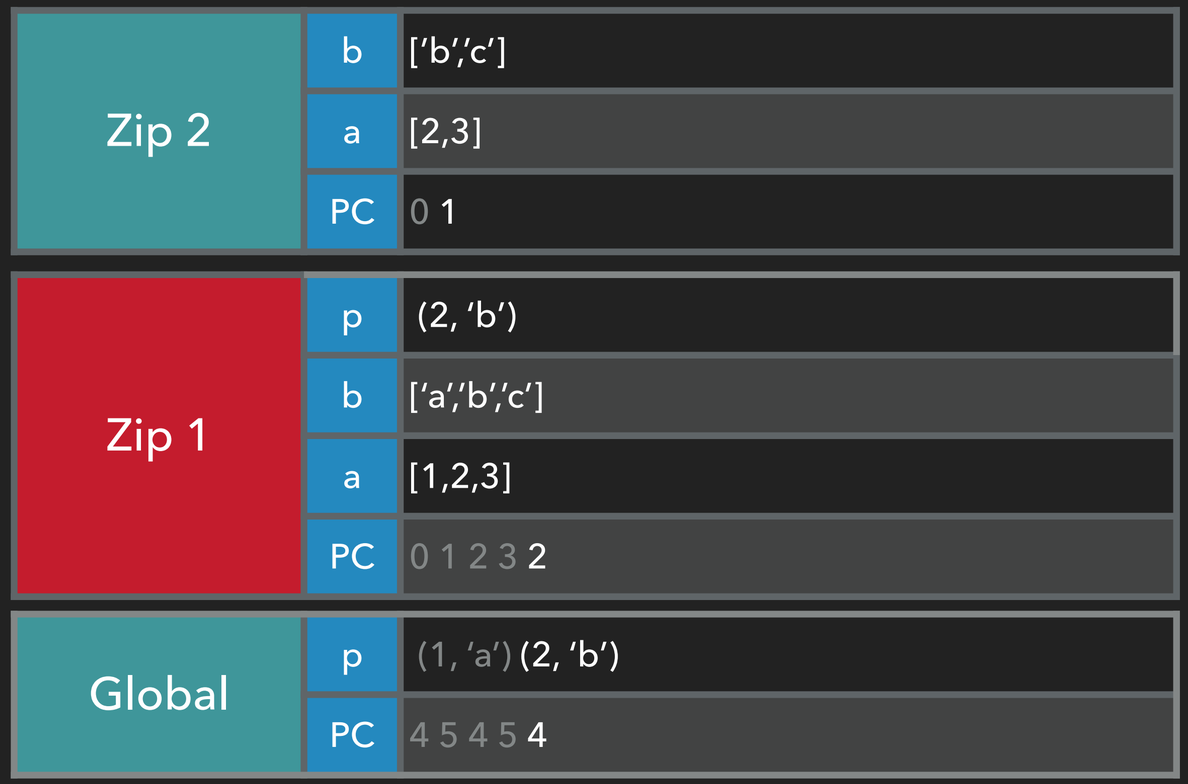
Zip 2	b	[‘b’,’c’]
	a	[2,3]
	PC	0 1
Zip 1	p	(2, ‘b’)
	b	[‘a’,’b’,’c’]
	a	[1,2,3]
	PC	0 1 2 3 2
Global	p	(1, ‘a’) (2, ‘b’)
	PC	4 5 4 5

```
   def zip(a, b):
0      if a and b:
1          yield (a[0], b[0])
2          for p in zip(a[1:], b[1:]):
3              yield p

4  for p in zip([1, 2, 3], ['a', 'b', 'c']):
5      print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
```



```
   def zip(a, b):
▸0     if a and b:
▸1         yield (a[0], b[0])
▸2         for p in zip(a[1:], b[1:]):
▸3             yield p

▸4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸5     print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
```

```
   def zip(a, b):
       if a and b:
0          yield (a[0], b[0])
1          for p in zip(a[1:], b[1:]):
2              yield p
3
4  for p in zip([1, 2, 3], ['a', 'b', 'c']):
5      print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
```

Zip 3	b	['c']
	a	[3]
	PC	0
Zip 2	p	-
	b	['b','c']
	a	[2,3]
	PC	0 1 2
Zip 1	p	(2, 'b')
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2 3 2
Global	p	(1, 'a') (2, 'b')
	PC	4 5 4 5 4

```
   def zip(a, b):
▸0     if a and b:
▸1         yield (a[0], b[0])
▸2         for p in zip(a[1:], b[1:]):
▸3             yield p

▸4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸5     print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
```

Zip 3	b	['c']
	a	[3]
	PC	0 1
Zip 2	p	-
	b	['b','c']
	a	[2,3]
	PC	0 1 2
Zip 1	p	(2, 'b')
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2 3 2
Global	p	(1, 'a') (2, 'b')
	PC	4 5 4 5 4

```
   def zip(a, b):
       if a and b:
0          yield (a[0], b[0])
1          for p in zip(a[1:], b[1:]):
2              yield p
3
4  for p in zip([1, 2, 3], ['a', 'b', 'c']):
5      print(p)
```

IMPRIME

```
> (1,’a’)
> (2,‘b’)
```

Zip 3	b	[‘c’]
	a	[3]
	PC	0 1
Zip 2	p	(3, ‘c’)
	b	[‘b’,‘c’]
	a	[2,3]
	PC	0 1 2 3
Zip 1	p	(2, ‘b’)
	b	[‘a’,‘b’,‘c’]
	a	[1,2,3]
	PC	0 1 2 3 2
Global	p	(1, ‘a’) (2, ‘b’)
	PC	4 5 4 5 4

```
   def zip(a, b):
▸ 0     if a and b:
▸ 1         yield (a[0], b[0])
▸ 2         for p in zip(a[1:], b[1:]):
▸ 3             yield p

▸ 4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸ 5     print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
```

Zip 3	b	['c']
	a	[3]
	PC	0 1
Zip 2	p	(3, 'c')
	b	['b','c']
	a	[2,3]
	PC	0 1 2 3 2
Zip 1	p	(2, 'b') (3, 'c')
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2 3 2 3
Global	p	(1, 'a') (2, 'b')
	PC	4 5 4 5 4

```
   def zip(a, b):
▸0     if a and b:
▸1         yield (a[0], b[0])
▸2         for p in zip(a[1:], b[1:]):
▸3             yield p

▸4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸5     print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
> (3,'c')
```

Zip 3	b	['c']
	a	[3]
	PC	0 1
Zip 2	p	(3, 'c')
	b	['b','c']
	a	[2,3]
	PC	0 1 2 3 2
Zip 1	p	(2, 'b') (3, 'c')
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2 3 2 3 2
Global	p	(1, 'a') (2, 'b') (3, 'c')
	PC	4 5 4 5 4 5

```
   def zip(a, b):
0      if a and b:
1          yield (a[0], b[0])
2          for p in zip(a[1:], b[1:]):
3              yield p

4  for p in zip([1, 2, 3], ['a', 'b', 'c']):
5      print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
> (3,'c')
```

Zip 3	b	['c']
	a	[3]
	PC	0 1
Zip 2	p	(3, 'c')
	b	['b','c']
	a	[2,3]
	PC	0 1 2 3 2
Zip 1	p	(2, 'b') (3, 'c')
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2 3 2 3 2
Global	p	(1, 'a') (2, 'b') (3, 'c')
	PC	4 5 4 5 4 5 4

```
   def zip(a, b):
▸ 0     if a and b:
▸ 1         yield (a[0], b[0])
▸ 2         for p in zip(a[1:], b[1:]):
▸ 3             yield p

▸ 4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸ 5     print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
> (3,'c')
```

Zip 3	b	['c']
	a	[3]
	PC	0 1
Zip 2	p	(3, 'c')
	b	['b','c']
	a	[2,3]
	PC	0 1 2 3 2
Zip 1	p	(2, 'b') (3, 'c')
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2 3 2 3 2
Global	p	(1, 'a') (2, 'b') (3, 'c')
	PC	4 5 4 5 4 5 4

```
   def zip(a, b):
       if a and b:
0          yield (a[0], b[0])
1          for p in zip(a[1:], b[1:]):
2              yield p
3

4  for p in zip([1, 2, 3], ['a', 'b', 'c']):
5      print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
> (3,'c')
```

Zip 3	b	['c']
	a	[3]
	PC	0 1 2
Zip 2	p	(3, 'c')
	b	['b','c']
	a	[2,3]
	PC	0 1 2 3 2
Zip 1	p	(2, 'b') (3, 'c')
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2 3 2 3 2
Global	p	(1, 'a') (2, 'b') (3, 'c')
	PC	4 5 4 5 4 5 4

```
   def zip(a, b):
       if a and b:
0          yield (a[0], b[0])
1          for p in zip(a[1:], b[1:]):
2              yield p
3
4  for p in zip([1, 2, 3], ['a', 'b', 'c']):
5      print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
> (3,'c')
```

Zip 4	b	None
	a	None
	PC	0
Zip 3	b	['c']
	a	[3]
	PC	0 1 2
Zip 2	p	(3, 'c')
	b	['b','c']
	a	[2,3]
	PC	0 1 2 3 2
Zip 1	p	(2, 'b') (3, 'c')
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2 3 2 3 2
	p	(1, 'a') (2, 'b') (3, 'c')

```
   def zip(a, b):
       if a and b:
0          yield (a[0], b[0])
1          for p in zip(a[1:], b[1:]):
2              yield p
3

4  for p in zip([1, 2, 3], ['a', 'b', 'c']):
5      print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
> (3,'c')
```

Zip 3	b	['c']
	a	[3]
	PC	0 1 2
Zip 2	p	(3, 'c')
	b	['b','c']
	a	[2,3]
	PC	0 1 2 3 2
Zip 1	p	(2, 'b') (3, 'c')
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2 3 2 3 2
Global	p	(1, 'a') (2, 'b') (3, 'c')
	PC	4 5 4 5 4 5 4

```
   def zip(a, b):
0      if a and b:
1          yield (a[0], b[0])
2          for p in zip(a[1:], b[1:]):
3              yield p

4  for p in zip([1, 2, 3], ['a', 'b', 'c']):
5      print(p)
```

IMPRIME

```
> (1,'a')
> (2,'b')
> (3,'c')
```

Zip 2	p	(3, 'c')
	b	['b','c']
	a	[2,3]
	PC	0 1 2 3 2
Zip 1	p	(2, 'b') (3, 'c')
	b	['a','b','c']
	a	[1,2,3]
	PC	0 1 2 3 2 3 2
Global	p	(1, 'a') (2, 'b') (3, 'c')
	PC	4 5 4 5 4 5 4

```
   def zip(a, b):
▸ 0     if a and b:
▸ 1         yield (a[0], b[0])
▸ 2         for p in zip(a[1:], b[1:]):
▸ 3             yield p

▸ 4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸ 5     print(p)
```

IMPRIME

```
> (1,’a’)
> (2,’b’)
> (3,‘c’)
```

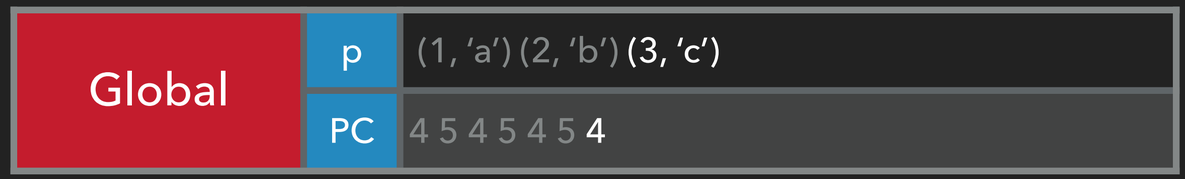
Zip 1	p	(2, ‘b’) (3, ‘c’)
	b	[‘a’,‘b’,‘c’]
	a	[1,2,3]
	PC	0 1 2 3 2 3 2
Global	p	(1, ‘a’) (2, ‘b’) (3, ‘c’)
	PC	4 5 4 5 4 5 4

```
   def zip(a, b):
▸0     if a and b:
▸1         yield (a[0], b[0])
▸2         for p in zip(a[1:], b[1:]):
▸3             yield p

▸4 for p in zip([1, 2, 3], ['a', 'b', 'c']):
▸5     print(p)
```

IMPRIME

```
> (1,’a’)
> (2,’b’)
> (3,‘c’)
```



PARTE A II

ZIP WITH

```
def zipWith(a, b, f):
    if a and b:
        yield f(a[0], b[0])
        for p in zipWith(a[1:], b[1:], f):
            yield p
```

PARTE B I

EJECUCION MISTERIO

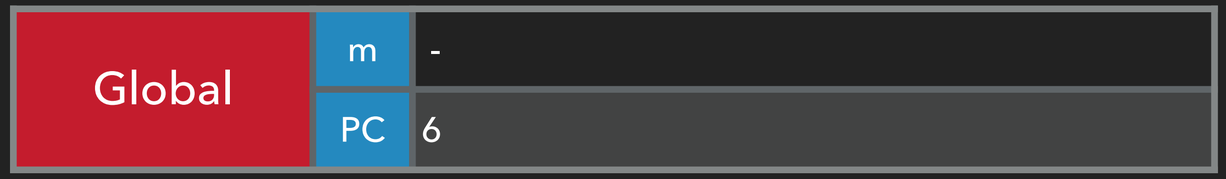
```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>
```



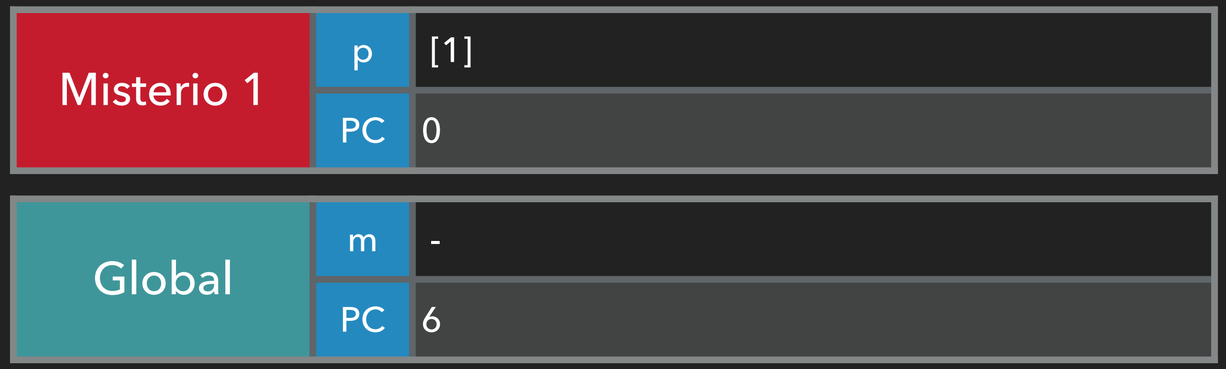
```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>
```



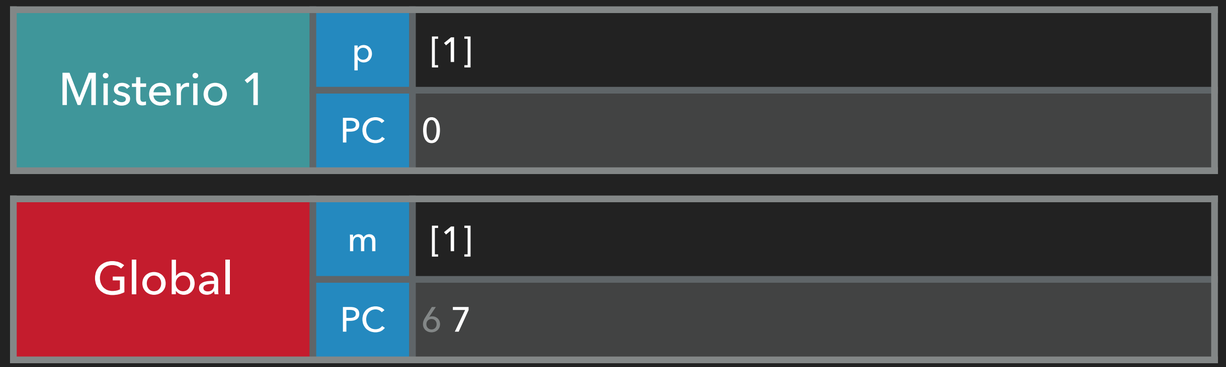
```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
```



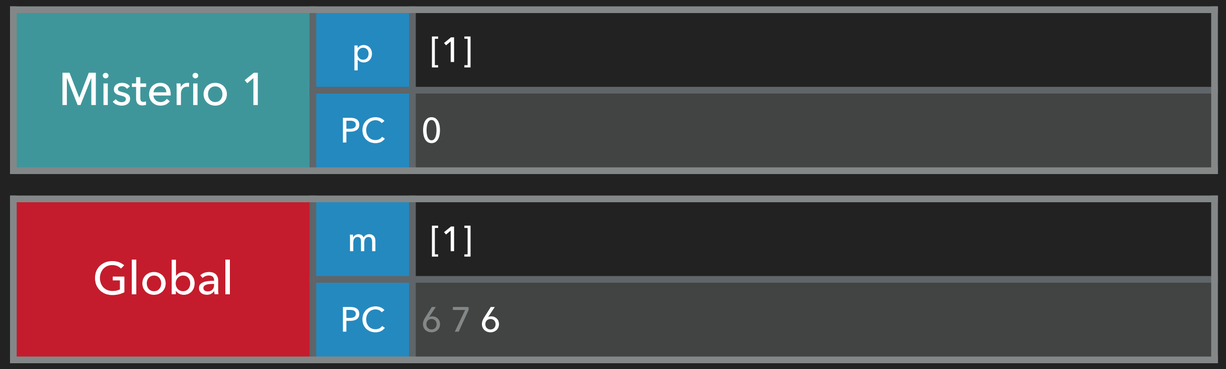
```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
```



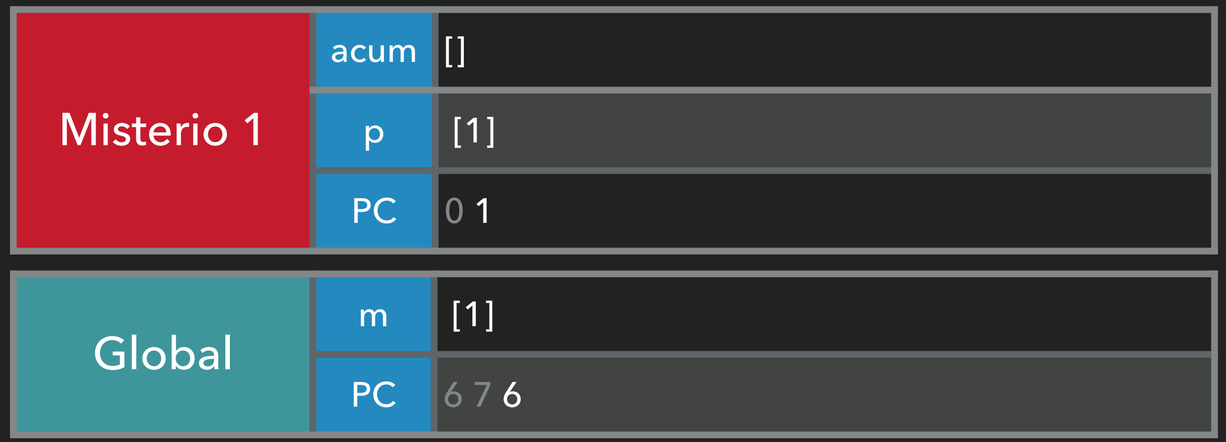
```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
```



```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

```
zipWith([0, *p], [*p, 0], lambda x, y: x + y)
# Genera la secuencia: 1, 1
```

IMPRIME

```
>[1]
```

Misterio 1	acum	[]
	p	[1]
	PC	0 1 2
Global	m	[1]
	PC	6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

```
zipWith([0, *p], [*p, 0], lambda x, y: x + y)
# Genera la secuencia: 1, 1
```

IMPRIME

Fast forward (termino el ciclo)

```
>[1]
```

Por alcance estatico, se restaura el valor de p

Misterio 1	m	-
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4
Global	m	[1]
	PC	6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
```

Misterio 2	p	[1, 1]
	PC	0
Misterio 1	m	-
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4
Global	m	[1]
	PC	6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
```

Misterio 2	p	[1, 1]
	PC	0
Misterio 1	m	[1,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5
Global	m	[1]
	PC	6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
```

Misterio 2	p	[1, 1]
	PC	0
Misterio 1	m	[1,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5
Global	m	[1] [1,1]
	PC	6 7 6 7

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
```

Misterio 2	p	[1, 1]
	PC	0
Misterio 1	m	[1,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5
Global	m	[1] [1,1]
	PC	6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
```

Misterio 2	p	[1, 1]
	PC	0
Misterio 1	m	[1,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4
Global	m	[1] [1,1]
	PC	6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

```
zipWith([0, *p], [*p, 0], lambda x, y: x + y)
# Genera la secuencia: 1, 2, 1
```

IMPRIME

```
>[1]
>[1,1]
```

Misterio 2	acum	[]
	p	[1, 1]
	PC	0 1 2
Misterio 1	m	[1,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4
Global	m	[1] [1,1]
	PC	6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

```
zipWith([0, *p], [*p, 0], lambda x, y: x + y)
# Genera la secuencia: 1, 2, 1
```

IMPRIME

Fast forward (termino el ciclo)

```
>[1]
>[1,1]
```

Por alcance estatico, se restaura el valor de p

Misterio 2	m	-
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4
Misterio 1	m	[1,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4
Global	m	[1] [1,1]
	PC	6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
```

Misterio 3	p	[1, 2, 1]
	PC	0
Misterio 2	m	-
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4
Misterio 1	m	[1,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4
Global	m	[1] [1,1]
	PC	6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
```

Misterio 3	p	[1, 2, 1]
	PC	0
Misterio 2	m	[1, 2, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5
Misterio 1	m	[1,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4
Global	m	[1] [1,1]
	PC	6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
```

Misterio 3	p	[1, 2, 1]
	PC	0
Misterio 2	m	[1, 2, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5
Misterio 1	m	[1,1] [1,2,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5
Global	m	[1] [1,1]
	PC	6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
>[1,2,1]
```

Misterio 3	p	[1, 2, 1]
	PC	0
Misterio 2	m	[1, 2, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5
Misterio 1	m	[1,1] [1,2,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5
Global	m	[1] [1,1] [1,2,1]
	PC	6 7 6 7 6 7

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
>[1,2,1]
```

Misterio 3	p	[1, 2, 1]
	PC	0
Misterio 2	m	[1, 2, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5
Misterio 1	m	[1,1] [1,2,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5
Global	m	[1] [1,1] [1,2,1]
	PC	6 7 6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
>[1,2,1]
```

Misterio 3	p	[1, 2, 1]
	PC	0
Misterio 2	m	[1, 2, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5
Misterio 1	m	[1,1] [1,2,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5 4
Global	m	[1] [1,1] [1,2,1]
	PC	6 7 6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
>[1,2,1]
```

Misterio 3	p	[1, 2, 1]
	PC	0
Misterio 2	m	[1, 2, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5 4
Misterio 1	m	[1,1] [1,2,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5 4
Global	m	[1] [1,1] [1,2,1]
	PC	6 7 6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

```
zipWith([0, *p], [*p, 0], lambda x, y: x + y)
# Genera la secuencia: 1, 3, 3, 1
```

IMPRIME

```
>[1]
>[1,1]
>[1,2,1]
```

Misterio 3	acum	[]
	p	[1, 2, 1]
	PC	0 1 2
Misterio 2	m	[1, 2, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5 4
Misterio 1	m	[1,1] [1,2,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5 4
Global	m	[1] [1,1] [1,2,1]
	PC	6 7 6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

```
zipWith([0, *p], [*p, 0], lambda x, y: x + y)
# Genera la secuencia: 1, 3, 3, 1
```

IMPRIME

Fast forward (termino el ciclo)

```
>[1]
>[1,1]
>[1,2,1]
```

Por alcance estatico, se restaura el valor de p

Misterio 3	m	-
	acum	[1, 3, 3, 1]
	p	[1, 2, 1] 1 3 3 1 [1, 2, 1]
	PC	0 1 2 3 2 3 2 3 2 3 4
Misterio 2	m	[1, 2, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5 4
Misterio 1	m	[1,1] [1,2,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5 4
Global	m	[1] [1,1] [1,2,1]
	PC	6 7 6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
>[1,2,1]
```

Misterio 4	p	[1, 3, 3, 1]
	PC	0
Misterio 3	m	-
	acum	[1, 3, 3, 1]
	p	[1, 2, 1] 1 3 3 1 [1, 2, 1]
	PC	0 1 2 3 2 3 2 3 2 3 4
Misterio 2	m	[1, 2, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5 4
Misterio 1	m	[1,1] [1,2,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5 4
Global	m	[1] [1,1] [1,2,1]
	PC	6 7 6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
>[1,2,1]
```

Misterio 4	p	[1, 3, 3, 1]
	PC	0
Misterio 3	m	[1, 3, 3, 1]
	acum	[1, 3, 3, 1]
	p	[1, 2, 1] 1 3 3 1 [1, 2, 1]
	PC	0 1 2 3 2 3 2 3 2 3 4 5
Misterio 2	m	[1, 2, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5 4
Misterio 1	m	[1,1] [1,2,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5 4
Global	m	[1] [1,1] [1,2,1]
	PC	6 7 6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
>[1,2,1]
```

Misterio 4	p	[1, 3, 3, 1]
	PC	0
Misterio 3	m	[1, 3, 3, 1]
	acum	[1, 3, 3, 1]
	p	[1, 2, 1] 1 3 3 1 [1, 2, 1]
	PC	0 1 2 3 2 3 2 3 2 3 4 5
Misterio 2	m	[1, 2, 1] [1, 3, 3, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5 4 5
Misterio 1	m	[1,1] [1,2,1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5 4
Global	m	[1] [1,1] [1,2,1]
	PC	6 7 6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
>[1,2,1]
```

Misterio 4	p	[1, 3, 3, 1]
	PC	0
Misterio 3	m	[1, 3, 3, 1]
	acum	[1, 3, 3, 1]
	p	[1, 2, 1] 1 3 3 1 [1, 2, 1]
	PC	0 1 2 3 2 3 2 3 2 3 4 5
Misterio 2	m	[1, 2, 1] [1, 3, 3, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5 4 5
Misterio 1	m	[1,1] [1,2,1] [1, 3, 3, 1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5 4 5
Global	m	[1] [1,1] [1,2,1]
	PC	6 7 6 7 6 7 6

```
    def misterio(p):
▸ 0     yield p
▸ 1     acum = []
▸ 2     for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
▸ 3         acum += [p]

▸ 4     for m in misterio(acum):
▸ 5         yield m


▸ 6 for m in misterio([1]):
▸ 7     print(m)
```

IMPRIME

```
>[1]
>[1,1]
>[1,2,1]
>[1,3,3,1]
```

Misterio 4	p	[1, 3, 3, 1]
	PC	0
Misterio 3	m	[1, 3, 3, 1]
	acum	[1, 3, 3, 1]
	p	[1, 2, 1] 1 3 3 1 [1, 2, 1]
	PC	0 1 2 3 2 3 2 3 2 3 4 5
Misterio 2	m	[1, 2, 1] [1, 3, 3, 1]
	acum	[1,2,1]
	p	[1, 1] 1 2 1 [1, 1]
	PC	0 1 2 3 2 3 2 3 4 5 4 5
Misterio 1	m	[1,1] [1,2,1] [1, 3, 3, 1]
	acum	[1, 1]
	p	[1] 1 1 [1]
	PC	0 1 2 3 2 3 4 5 4 5 4 5
Global	m	[1] [1,1] [1,2,1] [1,3,3,1]
	PC	6 7 6 7 6 7 6 7

Y ASÍ SUCESIVAMENTE

Los primeros 7 términos generados por misterio son:

```
>[1]
>[1,1]
>[1,2,1]
>[1,3,3,1]
>[1,4,6,4,1]
>[1,5,10,10,5,1]
>[1,6,15,20,15,6,1]
>. . .
```

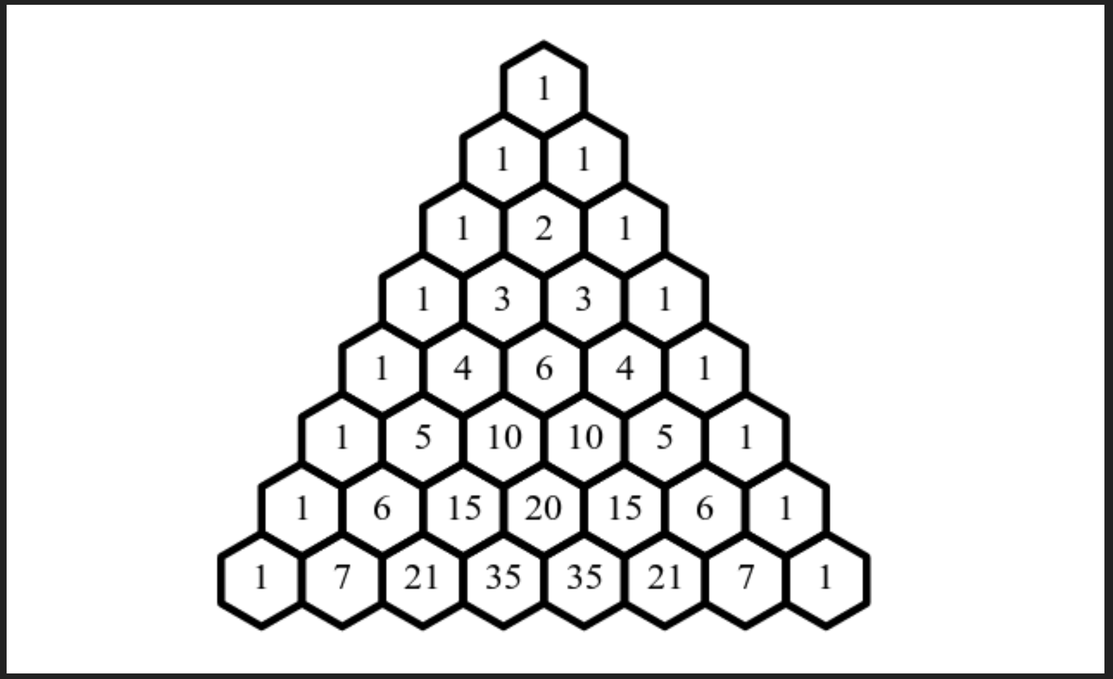
MISTERIO GENERARÁ INFINITOS TÉRMINOS

No existe ningún caso base que detenga las llamadas recursivas de misterio. La *linea 4* se va seguir ejecutando cada vez en niveles de recursion mas profundos sin detenerse. La única condición existente para que misterio se detenga es que zipWith no genere ningún valor, lo cual nunca se va cumplir, pues zipWith se esta evaluando siempre con arreglos de al menos un elemento *(pues se comienza con [1])* y por lo tanto, zipWith siempre genera al menos un valor dentro de misterio.

PARTE B II

RESOLVIENDO EL MISTERIO

MISTERIO GENERA LAS FILAS DEL TRIÁNGULO DE COEFICIENTES COMBINATORIOS




```
>[1]
>[1,1]
>[1,2,1]
>[1,3,3,1]
>[1,4,6,4,1]
>[1,5,10,10,5,1]
>[1,6,15,20,15,6,1]
>. . .
```

MISTERIO GENERA LAS FILAS DEL TRIANGULO DE COEFICIENTES COMBINATORIOS

- Haciendo uso de la propiedad de los coeficientes combinatorios.

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

- Utiliza el iterado zipWith para todos los elementos de la fila actual expresados como suma de la fila anterior.

```
zipWith([0, *p], [*p, 0], lambda x, y: x + y)
# Genera la fila actual en base a la anterior (p)
```

- zipWith le facilita el trabajo a misterio, pues se encarga de calcular todas las sumas en una sola expresión.

PARTE B III

SUSPENSO

```
def suspenso(p):
    for m in p:
        yield m
    acum = []
    for p in zipWith([0, *p], [*p, 0], lambda x, y: x + y):
        acum += [p]
    for m in suspenso(acum):
        yield m
```