Presentation 13

For-Loops

Announcements for This Lecture

Lessons

- **Videos 16.1-16.7** for today
- **Videos 17.1-17.5** next time

- **Prelim, 10/18 at 7:30 pm**
  - Material up to **TUESDAY**
  - Study guide is posted
- **Review TONIGHT**
  - **Zoom link on website**
  - **Will be recorded**

Assignments/Lab

- A3 is due **Tomorrow**
  - Survey is now posted
  - Will be graded before exam
- A4 posted on **Monday**
  - Longer time to do this one
  - Covers this lecture and next
- Work on Today's lab!
  - Lists will be on the exam
  - But not for-loops (today)

Activity Time: For-Loops

Function Definitions

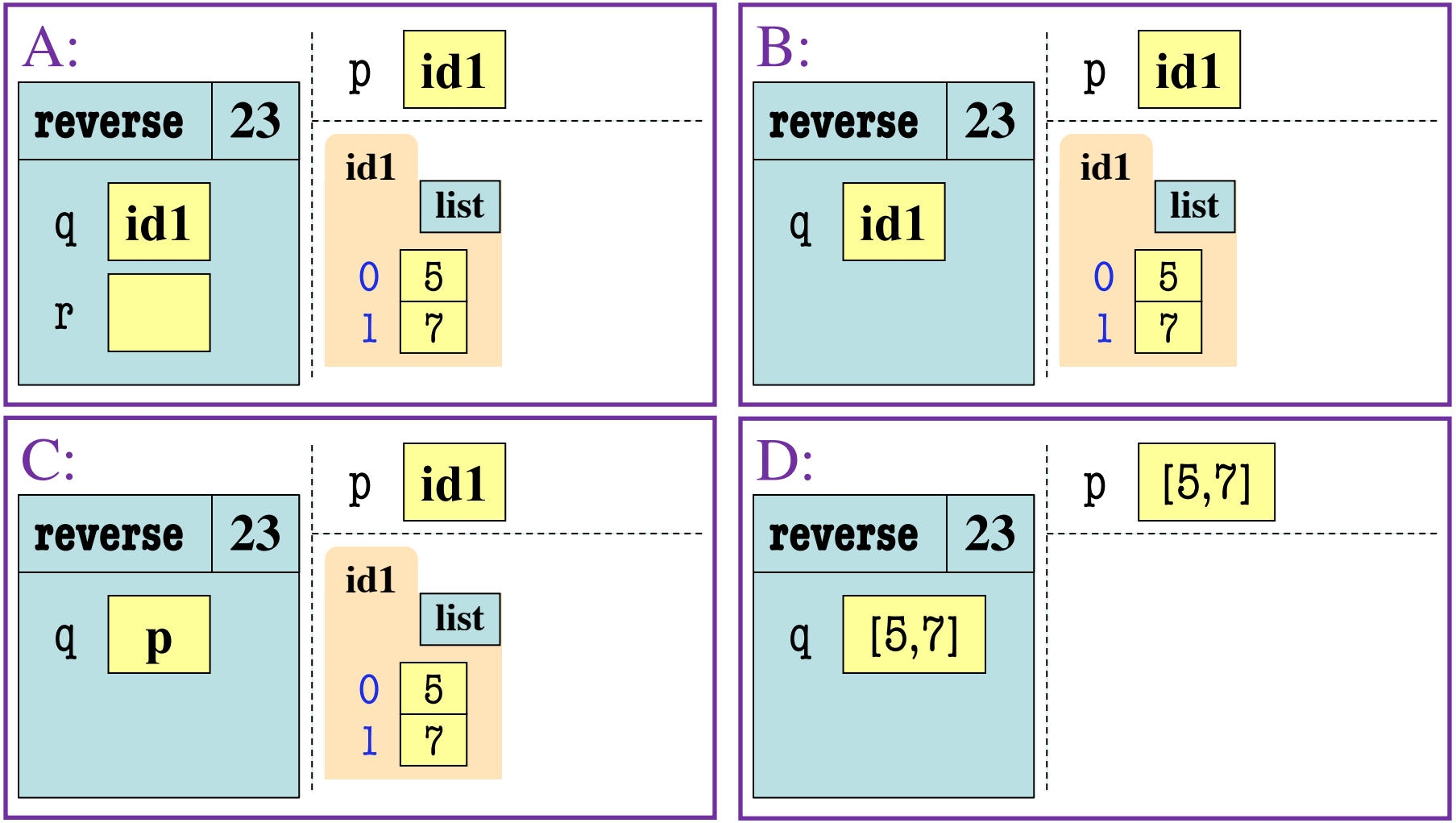
```
    def reverse(q):
23      r = []
24      for x in q:
25          r = [x]+r
26      return r
```

Function Call

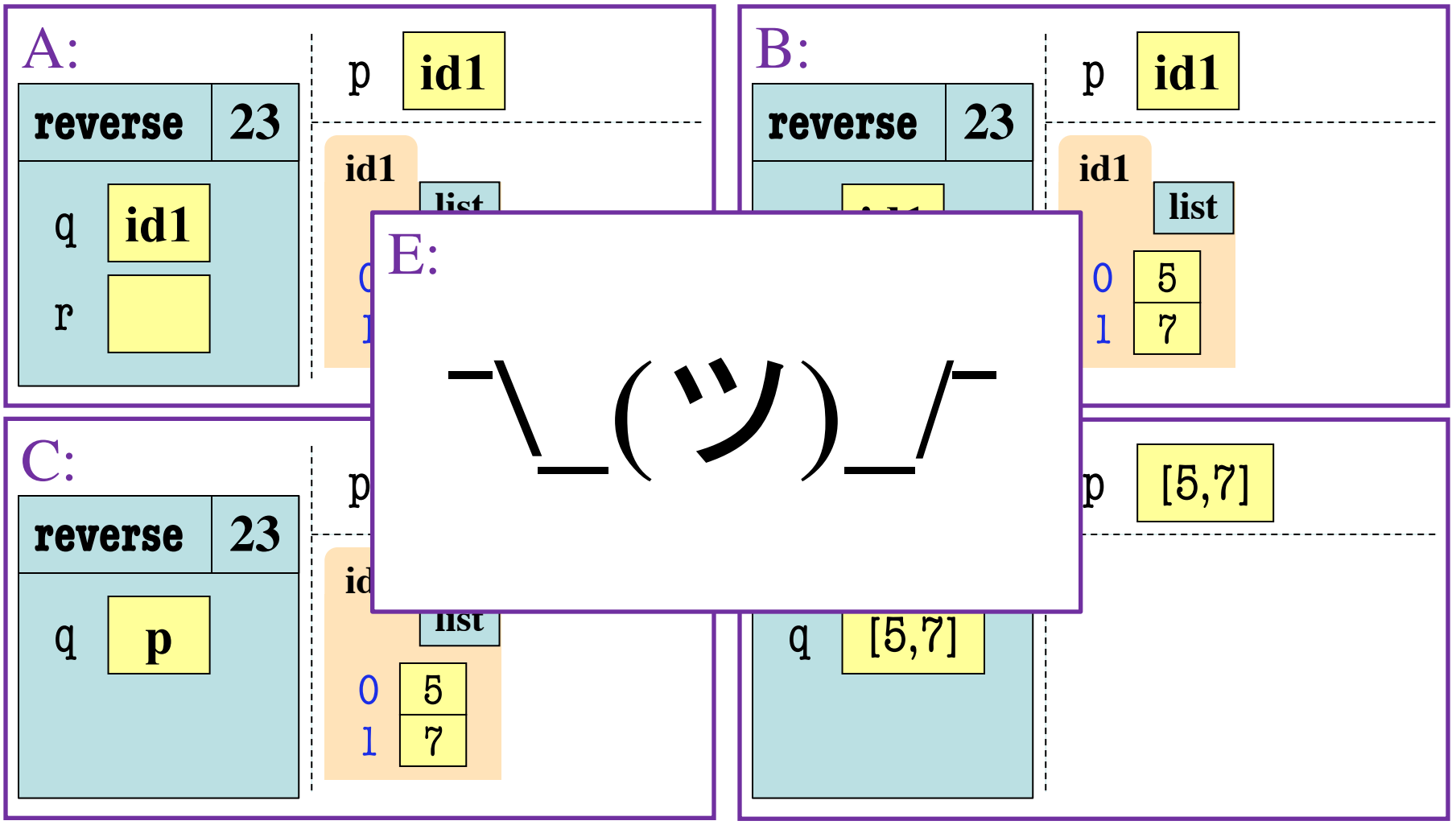
```
>>> p = [5,7]
>>> r = reverse(p)
```

What does memory look like at **start**?

Which One is Closest to Your Answer?



Which One is Closest to Your Answer?



Activity Time: For-Loops

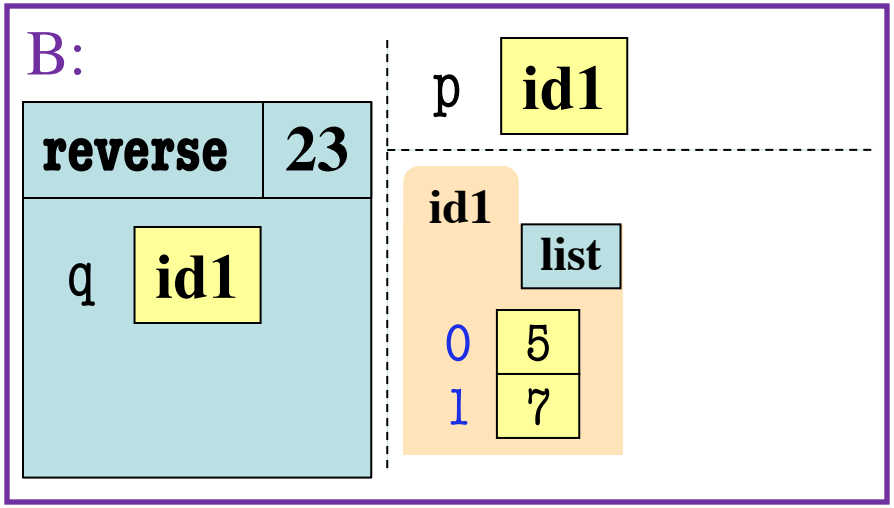
Function Definitions

```
def reverse(q):
23      r = []
24      for x in q:
25          r = [x]+r
26      return r
```

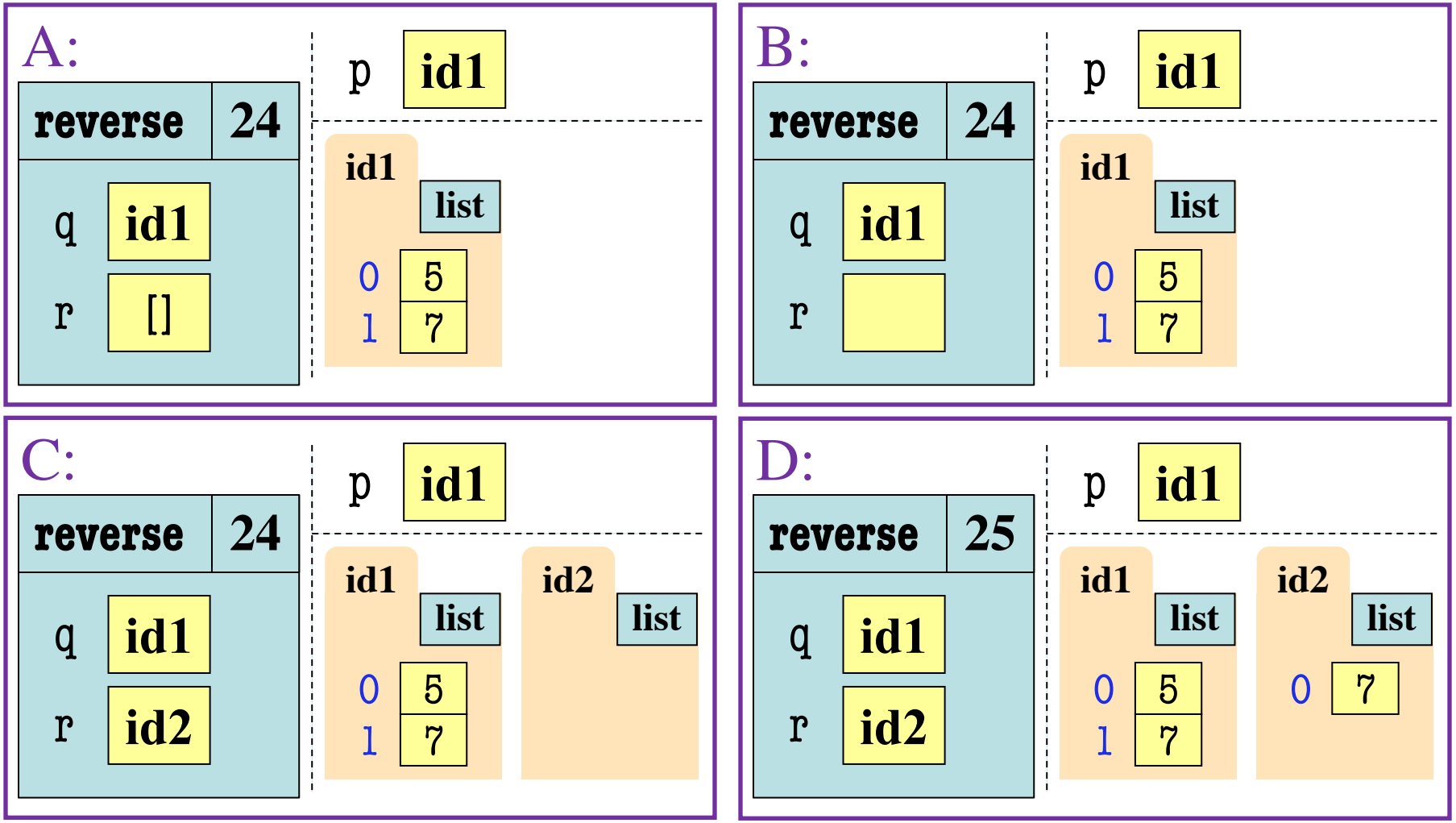
What is the **next step**?

Function Call

```
>>> p = [5,7]
>>> r = reverse(p)
```



Which One is Closest to Your Answer?



Activity Time: For-Loops

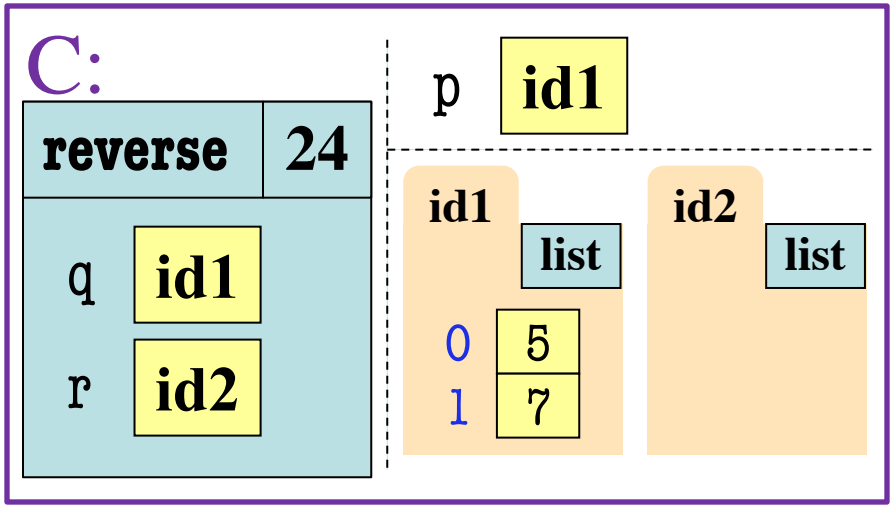
Function Definitions

```
def reverse(q):
23      r = []
24      for x in q:
25          r = [x]+r
26      return
```

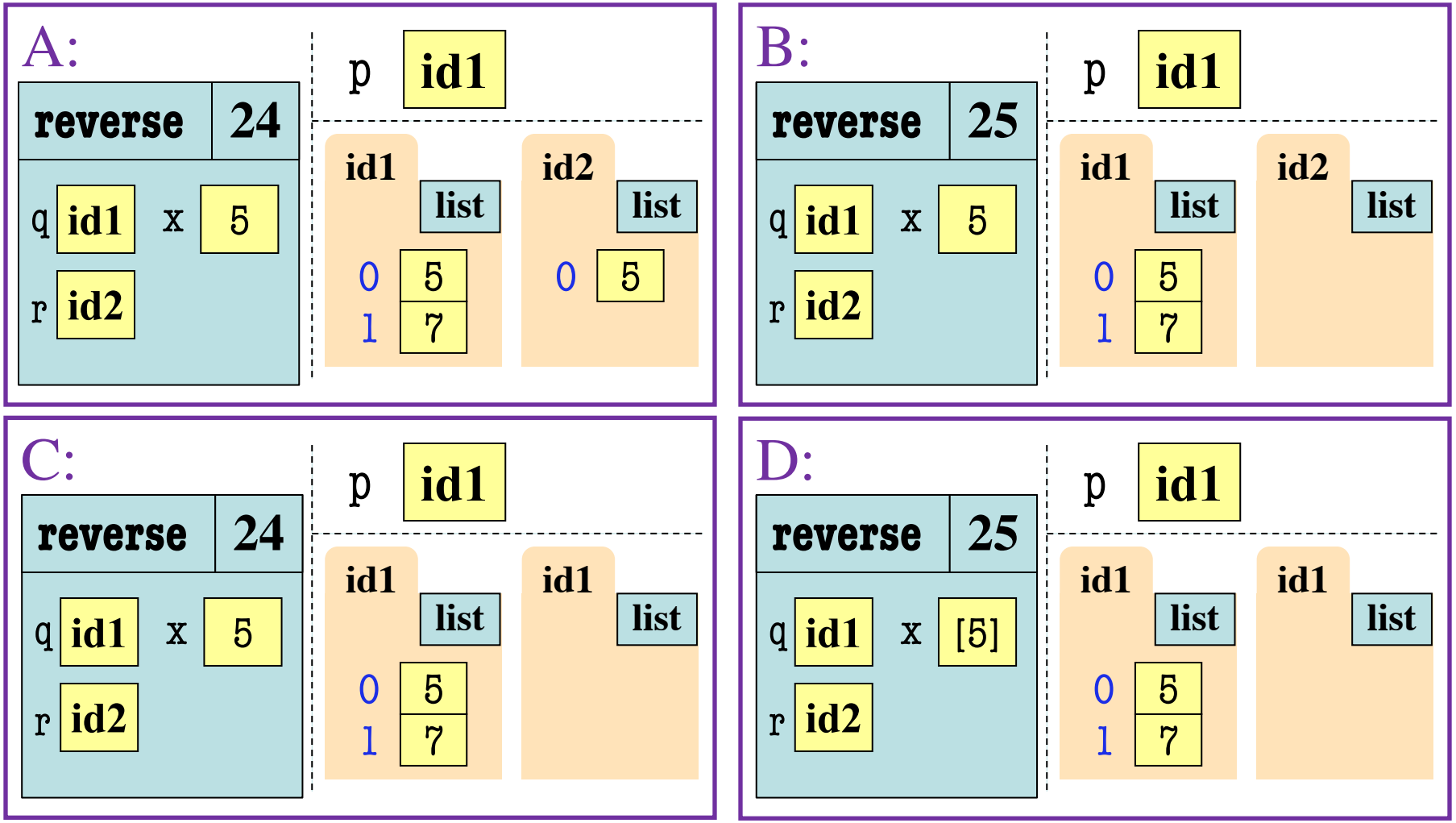
What is the **next step**?

Function Call

```
>>> p = [5,7]
>>> r = reverse(p)
```



Which One is Closest to Your Answer?



Activity Time: For-Loops

Function Definitions

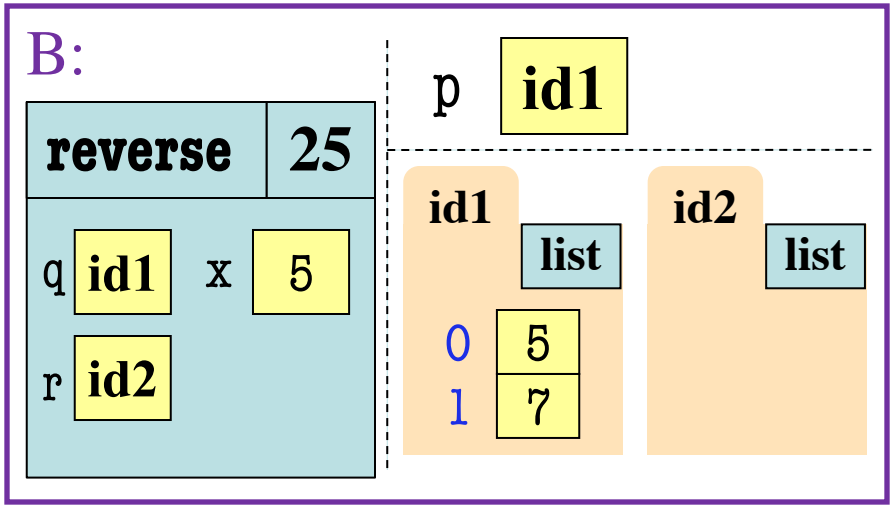
```
def reverse(q):
    r = []
    for x in q:
        r = [x]+r
    return
```

23 24 25 26

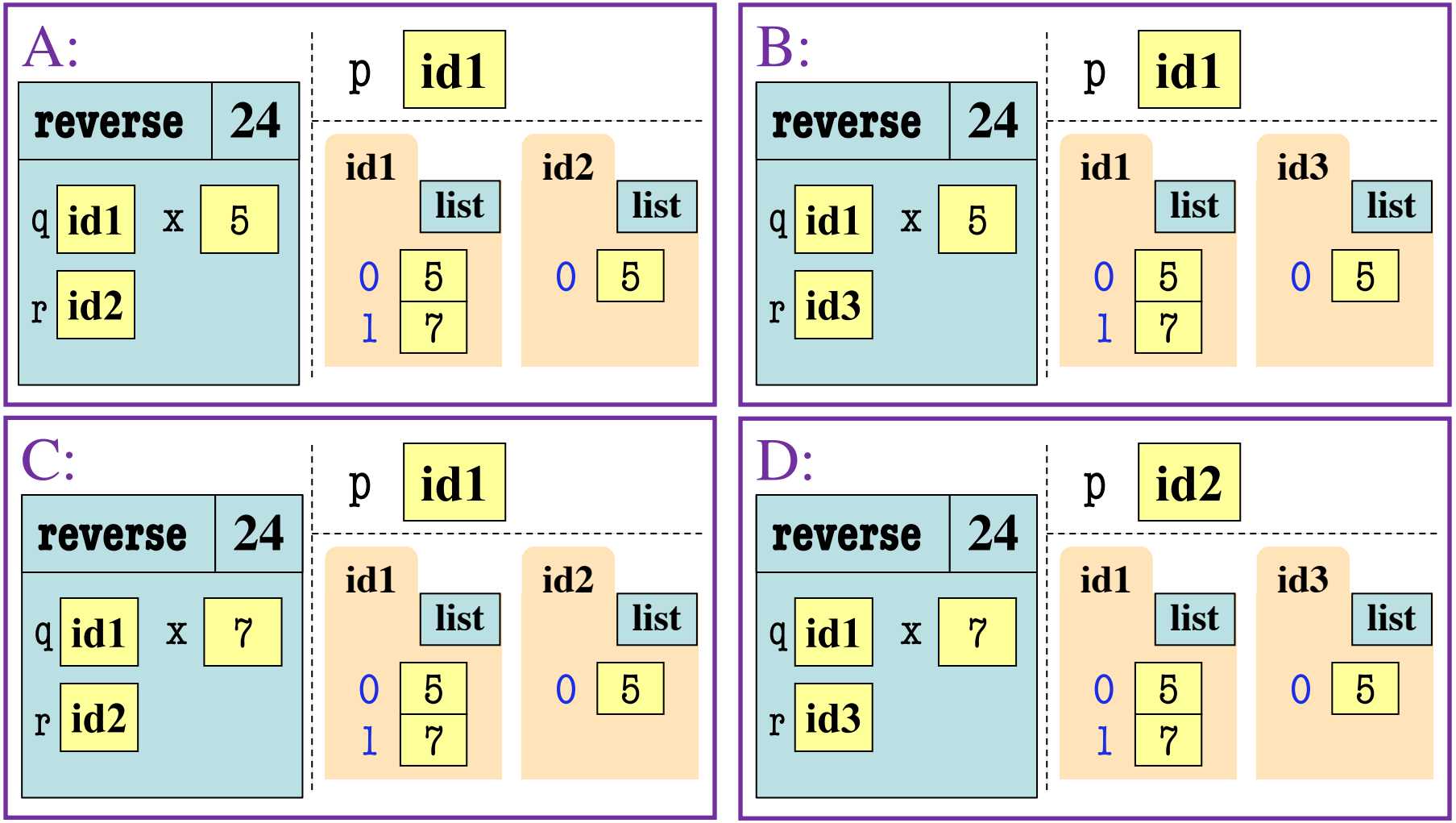
What is the **next step**?

Function Call

```
>>> p = [5,7]
>>> r = reverse(p)
```



Which One is Closest to Your Answer?



Activity Time: For-Loops

Function Definitions

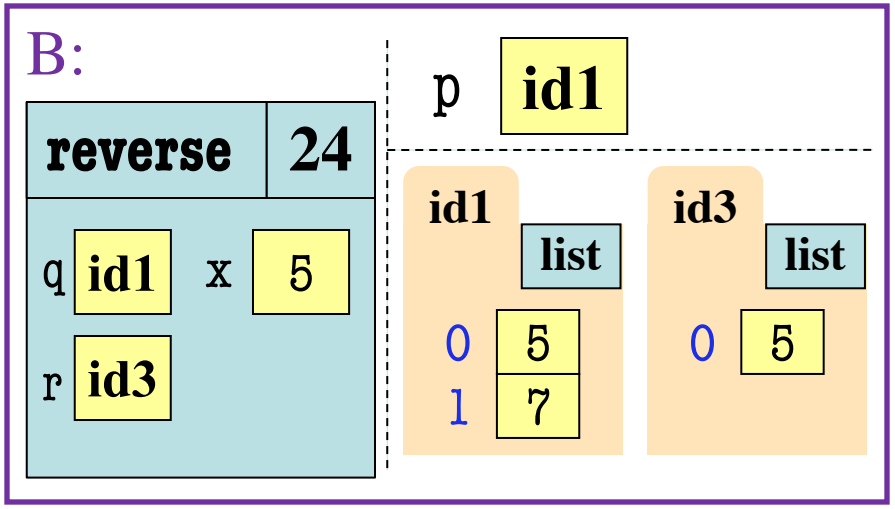
```
def reverse(q):
    r = []
    for x in q:
        r = [x]+r
    return
```

23 24 25 26

What is the **next step**?

Function Call

```
>>> p = [5,7]
>>> r = reverse(p)
```



Let's Skip Ahead One

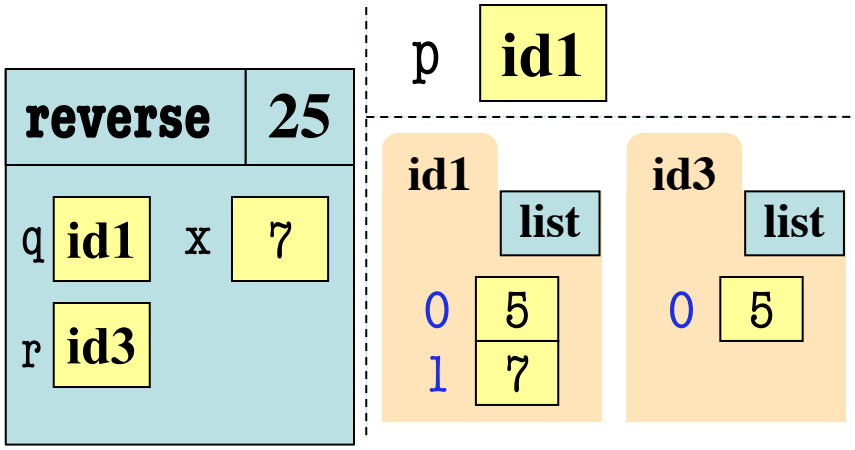
Function Definitions

```
    def reverse(q):
23      r = []
24      for x in q:
25          r = [x]+r
26      return
```

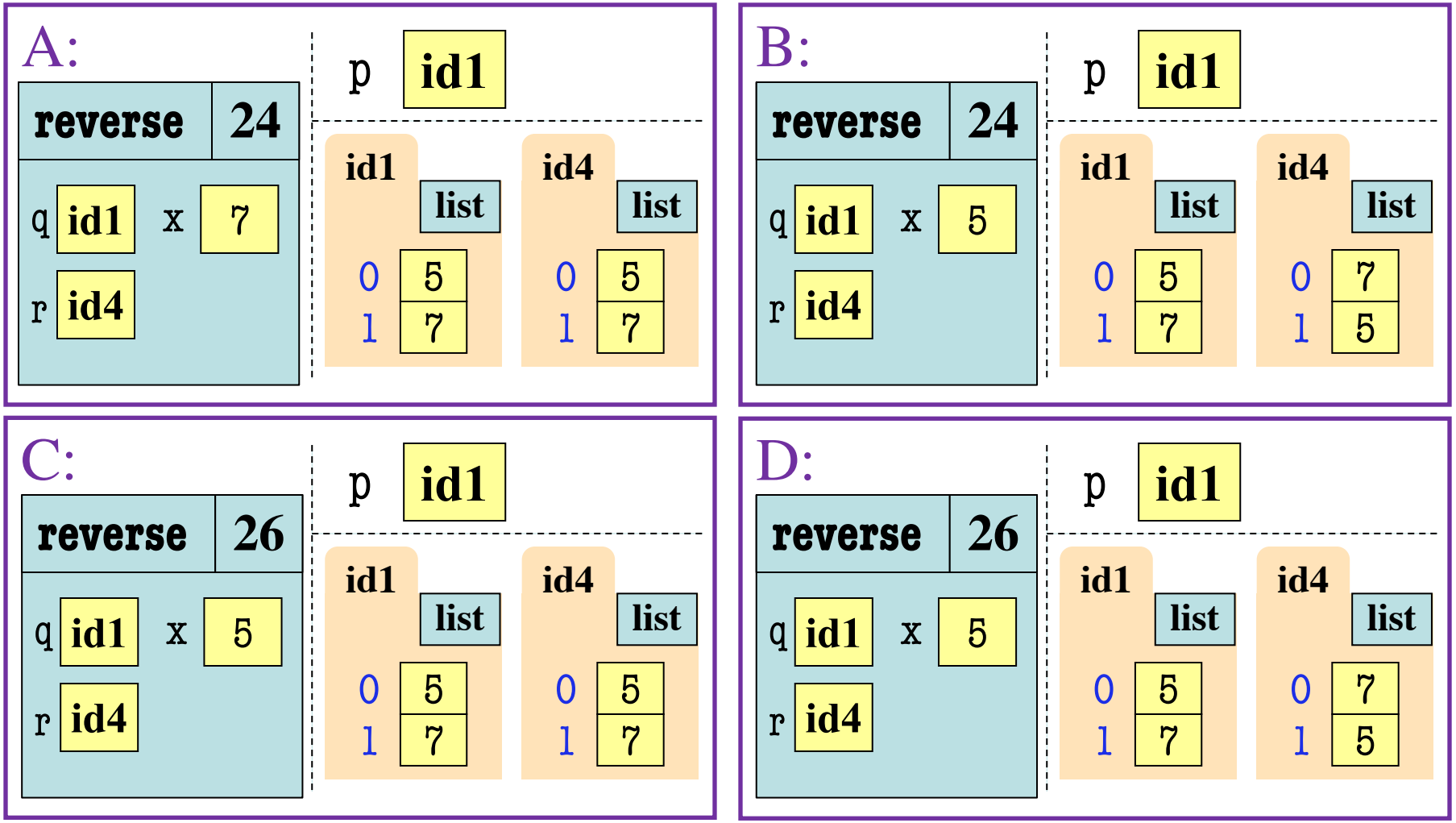
What is the **next step**?

Function Call

```
>>> p = [5,7]
>>> r = reverse(p)
```



Which One is Closest to Your Answer?



Let's Skip Ahead A Bit

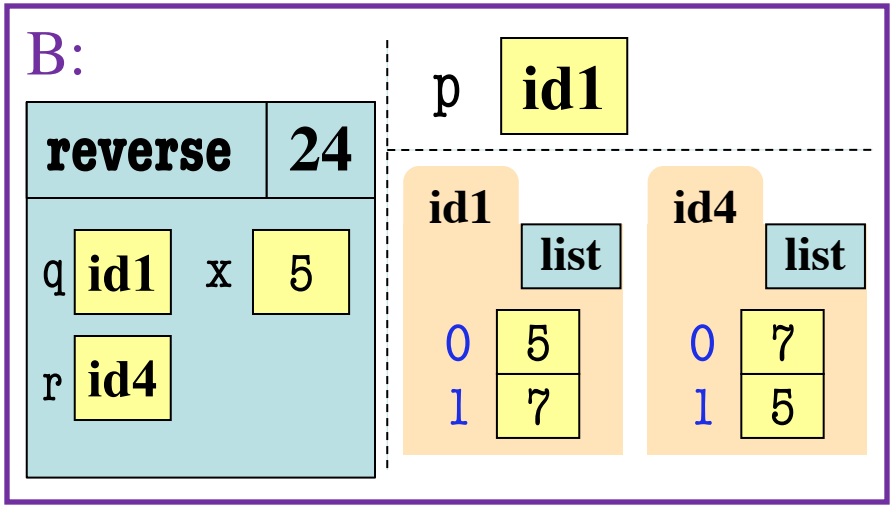
Function Definitions

```
    def reverse(q):
23      r = []
24      for x in q:
25          r = [x]+r
26      return
```

Still return when empty

Function Call

```
>>> p = [5,7]
>>> r = reverse(p)
```



Simple Accumulator

```
def avg(tup):
    """
    Returns average of all of the elements in the tuple.

    Remember that the average of a tuple is the sum of all of the
    elements in the tuple divided by the number of elements.

    Parameter tup: the tuple to check
    Precondition: tup is a tuple of numbers (int or float)
    """
    pass
```

Simple Accumulator

```
def avg(tup):
    """
    Returns average of all of the elements in the tuple.

    Remember that the average of a tuple is the sum of all of the
    elements in the tuple divided by the nu

    Parameter tup: the tuple to check
    Precondition: tup is a tuple of numbers
    """
    pass
```

Loop over?

A: Elements

B: Positions

C: Doesn't matter

Another Accumulator

```
def skip(s,n):
    """
    Returns a copy of s, only including positions that are multiples of n

    A position is a multiple of n if pos % n == 0.

    Examples: skip('hello world',1) returns 'hello world'
              skip('hello world',2) returns 'hlowrd'
              skip('hello world',3) returns 'hlwl'

    Parameter s: the string to copy
    Precondition: n is an int > 0
    """
    pass
```

Another Accumulator

```
def skip(s,n):
    """
    Returns a copy of s, only including positions that are multiples of n

    A position is a multiple of n if pos % n == 0.

    Examples: skip('hello world',1) returns 'hello wo
              skip('hello world',2) returns 'hlowrd'
              skip('hello world',3) returns 'hlwl'

    Parameter s: the string to copy
    Precondition: n is an int > 0
    """
    pass
```

Loop over?

A: Elements

B: Positions

C: Doesn't matter

No (???) Accumulator

```
def first_vowel(w):
    """
    Returns the position of the first vowel; -1 if there are no vowels.

    The vowels are 'a', 'e', 'i', 'o', 'u'.  The letter 'y' counts as vowel
    only if it is not the first element of w.

    Parameter w: the word to search
    Precondition: w is a nonempty string with only lowercase letters
    """
    pass
```

No (???) Accumulator

```
def first_vowel(w):
    """
    Returns the position of the first vowel; -1 if there are no vowels.

    The vowels are 'a', 'e', 'i', 'o', 'u'.  The letter 'y' counts as vowel
    only if it is not the first element of w.

    Parameter w: the word to search
    Precondition: w is a nonempty string w
    """
    pass
```

Loop over?

A: Elements

B: Positions

C: Doesn't matter

Mutable For-Loop

```
def reverse(alist):
    """
    Reverses the list in place

    Examples:
        If a = [1,2,2,3,4], reverse(a) changes a to [4,3,2,2,1]
        If a = [1,2,1], reverse(a) changes a to [1,2,1]
        If a = [], reverse(a) changes a to []

    Parameter alist: the list to modify
    Precondition: alist is a list of numbers (float or int)
    """
    pass
```

Mutable For-Loop

```
def reverse(alist):
    """
    Reverses the list in place

    Examples:
        If a = [1,2,2,3,4], reverse(a) changes a to [4,3,2,2,1]
        If a = [1,2,1], reverse(a) changes a to [1
        If a = [], reverse(a) changes a to []

    Parameter alist: the list to modify
    Precondition: alist is a list of numbers (floa
    """
    pass
```

Loop over?

A: Elements

B: Positions

C: Doesn't matter

Questions?