AP Review 1: Programming Problems (Part 1)

AP CSP REFERENCE SHEET

HERE is the **AP CSP Reference Sheet** that you **will have access to in the Digital AP Exams app on test day**, which shows all commands that might appear on the test in both text-based and block-based pseudocode.

Random, Selection (if, if-else), Iteration (loops), Procedures

Use the Reference Sheet as needed to help you answer the following questions!

Here is an algorithm written in AP Exam Text-based Pseudocode:

```
a ← 4
b ← 5
c ← 6 + a
a ← b
c ← c + b
DISPLAY(c)
```

What will be displayed when this algorithm is executed?	

Check!

Here is a command written in pseudocode: `RANDOM(4, 10)`	**a.** What's the **smallest** number that could get randomly selected? **b.** What's the **largest** number that ever gets randomly selected?

Check

Take a look at this algorithm:
 	
`x ← 6`
`y ← 10`
`DISPLAY(RANDOM(x, y))`
	a. How many *different* numbers are possible to be displayed?
	
	b. What's the likelihood (as a percentage) that the number displayed is a number greater than 8?

Check

Here's some AP Exam pseudocode:

```
randomNum ← RANDOM(0, 9)
if (randomNum < 4)
{
  DISPLAY("Less than 4!")
}
else
{
  DISPLAY("Not less than 4!")
}
```

What is the likelihood (as a percent) that `"Not less than 4!"` is displayed?

Check!

Multiple Choice

Peter wanted to write some code to choose a random number between 0 and 5, then print out whether the number was greater than 3, equal to 3, or less than three. Which of the following code segments would accomplish this?

Code Segment I

```
randomNum ← RANDOM(0, 5)
if (randomNum > 3)
{
  DISPLAY("Greater than 3!")
}
else
{
  if (randomNum = 3)
  {
    DISPLAY("Equal to 3!")
  }
  else
  {
    DISPLAY("Less than 3!")
  }
}
```

Code Segment II

```
randomNum ← RANDOM(0, 5)
if (randomNum > 3)
{
  DISPLAY("Greater than 3!")
}
if (randomNum = 3)
{
  DISPLAY("Equal to 3!")
}
if (randomNum < 3)
{
  DISPLAY("Less than 3!")
}
```

A. Code segment I only
B. Code segment II only
C. Both code segments
D. Neither code segment

Answer:

Check my answer!

Multiple choice:

Which of the following will return an *even* random number between 1 and 10?

```
A)  RANDOM(1, 5)
B)  RANDOM(2, 10)
C)  RANDOM(1, 10) / 2
D)  RANDOM(1, 5) * 2
```

Check!

On the AP exam, functions are called "procedures" but they work the same way.

Here is a procedure (function) named `tripler` that has one *parameter* named `number` and that `RETURN`s a value:

```
PROCEDURE tripler(number)
{
  tripled ← number * 3
  RETURN tripled
}
```

And here is an algorithm in which the procedure gets *called* (**in bold**):

```
DISPLAY("Please enter a number:")
userNum ← INPUT()
result ← tripler(userNum)
DISPLAY("Your number tripled is: " + result)
```

What will be displayed when the calling algorithm gets executed if the the user inputs the value "**20**" when prompted:	

Check answer

Below is a `mystery` procedure written in AP Exam pseudocode, both in **block-based** (left) and in **text-based** (right) pseudocode; note that both representations are of the *same* procedure.

Block-based:

```
PROCEDURE mystery [number]
    REPEAT UNTIL (number > 0)
        [number ← number + 4]
    IF (number = 2)
        [number ← number * 2]
    [RETURN number * 4]
```

Text-based:

```
PROCEDURE mystery (number)
{
   REPEAT UNTIL (number > 0)
   {
      number ← number + 4
   }
   IF (number = 2)
   {
      number ← number * 2
   }
   RETURN number * 4
}
```

Determine the value that gets returned by each of the following calls to the procedure:

Block-based:	**Text-based:**
mystery [-15]	Mystery (-15) =
mystery [-10]	Mystery (-10) =
mystery [-9]	Mystery (-9) =
mystery [-8]	Mystery (-8) =
mystery [-2]	Mystery (-2) =
mystery [0]	Mystery (0) =
mystery [2]	Mystery (2) =
mystery [3]	Mystery (3) =
mystery [5]	Mystery (5) =
mystery [10]	Mystery (10) =

Confirm your answers!

Here is a procedure named `mystery` that has a list of numbers as a parameter:

```
PROCEDURE mystery(num_list)
{
  count = 0
  FOR EACH num IN num_list
  {
    IF num > 5
    {
      DISPLAY(num)
      count ← count + 1
    }
  }
  DISPLAY("end!")
  RETURN count
}
```

What gets displayed when the following code is executed?

```
result ← mystery([1, 6, 2, 7, 3, 8, 4, 9, 5])
DISPLAY("the result is: " + result)
```

What gets displayed?	*check answer*

When a return statement is reached in a procedure, the procedure ends immediately and the value gets returned. No more code inside the procedure executes. If the return happens inside a loop, the loop also ends.

Here is a similar procedure named `mystery` that has a list of numbers as a parameter:

```
PROCEDURE mystery(num_list)
{
  count = 0
  FOR EACH num IN num_list
  {
    IF num > 5
    {
      DISPLAY(num)
      count ← count + 1
      RETURN count
    }
  }
  DISPLAY("end!")
  RETURN count
}
```

What gets displayed when the following code is executed?

```
result ← mystery([1, 6, 2, 7, 3, 8, 4, 9, 5])
DISPLAY("the result is: " + result)
```

What gets displayed?	*check answer*

AP EXAM PRACTICE QUESTION

Which of the following statements about `PROCEDURE Mystery` are true?

```
PROCEDURE Mystery (number)
{
   REPEAT UNTIL (number ≥ 0)
   {
      number ← number + 2
   }
   IF (number = 0)
   {
     RETURN true
   }
   ELSE
   {
     RETURN false
   }
}
```

I. The procedure will return `true` for any negative, even integer.
II. The procedure will return `false` for any positive integer.
III. The procedure will return `false` for any odd integer, either positive and negative.

	(A) I only
	(B) II only
	(C) I and II only
	(D) I, II, and III

Check your answer!

Four similar algorithms are written in block-based AP Exam pseudocode below. ***Pay careful attention to how they differ!*** **Note:** the lines that say `i ← i + 1` use both i and 1

a.

```
i ← 0
sum ← 0
REPEAT UNTIL (i = 5)
    sum ← sum + i
    i ← i + 1
DISPLAY [sum]
```

What value will be displayed?

b.

```
i ← 0
sum ← 0
REPEAT UNTIL (i > 5)
    i ← i + 1
    sum ← sum + i
DISPLAY [sum]
```

What value will be displayed?

c.

```
i ← 0
sum ← 0
REPEAT UNTIL (i = 5)
    i ← 0
    sum ← sum + i
    i ← i + 1
DISPLAY [sum]
```

This loop gets stuck going forever and ever and the sum is never displayed! This is called an infinite loop. Explain why this happens:

d.

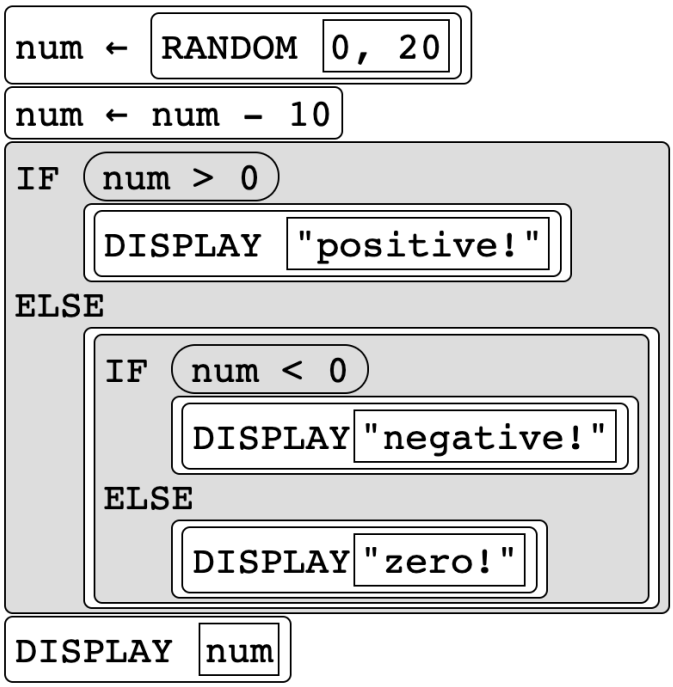
```
i ← 0
sum ← 0
REPEAT UNTIL (i = 5)
    sum ← sum + i
    i ← i + 2
DISPLAY [sum]
```

Is this also an **infinite loop**? If **not**, determine its output:

Confirm answers & explanations

An algorithm is written below in both **text-based** and equivalent **block-based** pseudocode:

```
num ← RANDOM (0, 20)
num ← num - 10
IF (num > 0)
{
   DISPLAY ("positive!")
}
ELSE
{
   IF (num < 0)
   {
      DISPLAY ("negative!")
   }
   ELSE
   {
      DISPLAY ("zero!")
   }
}
DISPLAY (num)
```



What **two** words will be displayed if the random number selected in the first line is **6**?	
What **two** words will be displayed if the random number selected in the first line is **18**?	
What **two** words will be displayed if the random number selected in the first line is **10**?	

Check your answers

Lists & List Operations

Don't forget! FOR LISTS ON THE AP EXAM, THE FIRST INDEX IS 1, NOT 0!

Use the Reference Sheet to help answer the following problems involving lists and list operations!

Here is some code written in AP Exam pseudocode that involves a **list**:

```
twoLetterWords ← ["ok", "it", "so", "pa", "no"]
num ← LENGTH(twoLetterWords) * 6
DISPLAY(num)
```

What will be displayed when this code is executed?	

Check!

Here is some code written in AP Exam pseudocode that involves a **list** and a **for each** loop:

```
numberList ← [10, -3, 5, 0, -7, 4, 8, -6]
FOR EACH item IN numberList
{
  IF (item > 0)
  {
    DISPLAY(item)
  }
}
```

What will be displayed when this code is executed?	

Check!

Here is some code written in AP Exam pseudocode that involves a **list** and a **for each** loop:

```
numberList ← [5, 7, 4, 10, 6, 2]
num1 ← numberList[2]
DISPLAY(num1)
DISPLAY(numberList[4])
```

What will be displayed when this code is executed?	

Check!

WARM UP QUESTION

Below is a variable `animals` that stores a list:

```
animals ← ["cat", "fish", "lizard", "giraffe", "dog"]
```

What is the value of `animals[4]`?

X the correct answer choice.

	A. `"m"`
	B. `"fish"`
	C. `"giraffe"`
	D. `5`

Check answer

WARM UP QUESTION

Below is a variable `animals` that stores a list:

```
animals ← ["cat", "fish", "lizard", "giraffe", "dog"]
```

What is the value of `LENGTH(animals)`?

X the correct answer choice.

	A. `5`
	B. `7`
	C. `[3, 4, 6, 7, 3]`
	D. `None of the above`

Check answer

AP EXAM PRACTICE QUESTION

Selena wrote the following code:

```
m ← 2
n ← 1
subjects ← ["math", "ELA", "AP CSP", "science"]
mystery ← subjects[n]
REPEAT m TIMES
{
    n ← n + 1
    mystery ← subjects[n]
}
DISPLAY(mystery)
```

What value is displayed?

X the correct answer choice.

	A. "math"
	B. "AP CSP"
	C. "science"
	D. ["math", "ELA", "AP CSP"]

Check answer

AP EXAM PRACTICE QUESTION

A programmer wrote the following program to determine the **average** of a list of numbers called `numberList`:

```
numberList ← [1, 2, 3, 4]
sum ← 0
FOR EACH number IN numberList
    sum ← sum + number
DISPLAY sum / LENGTH [numberList]
```

If the program displays 2.5, which statement is **true**?

X the correct answer choice.

	A. The result is correct, and the single test case with numberList = [1, 2, 3, 4] is sufficient for concluding that the program will work for all possible lists of numbers
	B. The result is correct, but the single test case where numberList = [1, 2, 3, 4] is not sufficient for concluding that the program will work for all other lists of numbers, and the programmer should test additional numberLists.
	C. The result is correct, but the program only works for numberLists that contain exactly four elements.
	D. The result is incorrect, so the program has an error in it that the programmer should find and debug.

Check my answer

AP EXAM PRACTICE QUESTION

A programmer wrote the following procedure to determine the average of a list of numbers called `numberList`:

```
Line 1:  PROCEDURE CalculateAverage(numberList)
Line 2:  {
Line 3:     sum ← 0
Line 4:     count ← 1
Line 5:     FOR EACH number IN numberList
Line 6:     {
Line 7:        sum ← sum + number
Line 8:        count ← count + 1
Line 9:     }
Line 10:    average ← sum / count
Line 11:    DISPLAY (average)
Line 12: }
```

This procedure will not correctly calculate the average; which change to the code will fix the error?

X the correct answer choice.

	A. Change line 3 to `sum ← numberList[1]`
	B. Change line 4 to `count ← 0`
	C. Interchange line 7 and line 8
	D. Change line 11 to `DISPLAY (average - 1)`

Check my answer

Don't forget! FOR LISTS ON THE AP EXAM, THE FIRST INDEX IS 1, NOT 0!

From the Reference Sheet, here are what **INSERT**, **APPEND**, and **REMOVE** do:

Text:
`INSERT(aList, i, value)`
Block:
`INSERT aList, i, value`
	Any values in `aList` at indices greater than or equal to `i` are shifted one position to the right. The length of the list is increased by 1, and `value` is placed at index `i` in `aList`.

Text:
`APPEND(aList, value)`
Block:
`APPEND aList, value`
	The length of `aList` is increased by 1, and `value` is placed at the end of `aList`.

Text:
`REMOVE(aList, i)`
Block:
`REMOVE aList, i`
	Removes the item at index `i` in `aList` and shifts to the left any values at indices greater than `i`. The length of `aList` is decreased by 1.

AP EXAM PRACTICE QUESTION

Which of the following algorithms output the value 3?

Algorithm I

```
grades ← []
APPEND(grades, 2)
APPEND(grades, 3)
APPEND(grades, 1)
DISPLAY(grades[2])
```

Algorithm II

```
numbers ← [3, 1]
APPEND(numbers, 2)
sum ← numbers[2] + numbers[3]
DISPLAY(sum)
```

A. Algorithm I only
B. Algorithm II only
C. Both Algorithms I and II
D. Neither Algorithm I nor II

My answer (A, B, C, or D):	*After checking answer below, the correct answer is:*

Check your answer!

AP EXAM PRACTICE QUESTION

Which of the following algorithms output the value `29`?

Algorithm I

```
temps ← [31, 30, 29]
REMOVE(temps, 2)
DISPLAY(temps[2])
```

Algorithm II

```
numbers ← []
APPEND(numbers, 30)
APPEND(numbers, 29)
REMOVE(numbers, 2)
value ← numbers[1] - LENGTH(numbers)
DISPLAY(value)
```

A. Algorithm I only
B. Algorithm II only
C. Both Algorithms I and II
D. Neither Algorithm I nor II

My answer (A, B, C, or D):		*After checking answer, the correct answer is:*	

Check your answer!

AP EXAM PRACTICE QUESTION

Here is some code that executes:

```
num ← 4
mysteryList ← [num, 3]
APPEND(mysteryList, 7)
INSERT(mysteryList, 1, 2)
INSERT(mysteryList, 1, num)
INSERT(mysteryList, num, 5)
DISPLAY(mysteryList)
```

Which list accurately reflects the `mysteryList` that gets displayed at the end?

A. `[4, 4, 2, 3, 5, 7]`
B. `[4, 1, 3, 4, 4, 7]`
C. `[4, 2, 4, 5, 3, 7]`
D. None of the above

AP Exam Pro Tip: *Keep track of how mysteryList is changing by* ***writing*** *it down!*

My answer (A, B, C, or D):		*After checking answer, the correct answer is:*	

Check your answer!

AP EXAM PRACTICE QUESTION

Here is some code that executes:

```
cakeIngredients ← ["butter", "flour", "eggs", "oil"]
cakeIngredients[2] ← "sugar"
cakeIngredients[4] ← cakeIngredients[3]
DISPLAY(cakeIngredients)
```

Which list accurately reflects the `cakeIngredients` that gets displayed at the end?

A. `["butter", "flour", "eggs", "eggs"]`
B. `["butter", "sugar", "eggs", "eggs"]`
C. `["butter", "sugar", "oil", "oil"]`
D. `["butter", "sugar", "eggs", "oil"]`

My answer (A, B, C, or D):	*After checking answer, the correct answer is:*

Check your answer!

Using Trace Tables

Using a **trace table** can be helpful when analyzing algorithms that involve **iteration** (loops)!

Here is an algorithm written in AP exam pseudocode:

```
a ← 3
b ← 1
REPEAT 5 TIMES
{
  a ← a + b
  b ← b + 1
}
DISPLAY(a + b)
```

a. Complete the trace table to help you determine what gets displayed at the end.

Started for you:

a	b	Iteration #
3	1	
		1
		2
		3
		4
		5

b. What will be displayed?

Check answers

Here is an algorithm written in AP exam pseudocode:

```
numList ← [8, 7, 9, 5]
len ← LENGTH(numList)
count ← 1
sum ← 0

REPEAT len TIMES
{
  sum ← sum + numList[count]
  count ← count + 1
}

DISPLAY(sum + count)
```

ON SCRAP PAPER, create a trace table to help you determine what will be displayed when this algorithm is executed; you want to track all variables as they change!

What gets displayed?

Compare Solution with video

Here is an algorithm written in AP exam pseudocode:

```
num ← 10
output ← 5
REPEAT UNTIL (num < 0)
    IF (num > output)
        output ← output + num
    ELSE
        output ← output - num
    num ← num - 2
DISPLAY output
```

ON SCRAP PAPER, complete the trace table to help you determine what will be displayed when this algorithm is executed:

num	output

What gets displayed?

Compare Solution with video

Here is an algorithm in pseudocode:

```
nums ← [2, 5, 7]
INSERT(nums, 2, 9)
APPEND(nums, 3)
APPEND(nums, 8)
APPEND(nums, 4)
nums[3] ← 1
REMOVE(nums, 1)
INSERT(nums, 5, 6)
DISPLAY(nums)
```

ON SCRAP PAPER, complete the trace table (started for you) to help you determine what gets printed:

nums
[2, 5, 7]

What gets displayed?

Compare Solution

What gets displayed when the following code segment is executed?

```
nums ← [3, 1, 6, 4]
INSERT(nums, 3, 5)
APPEND(nums, 2)
nums[4] ← 1
REMOVE(nums, 2)
x ← nums[3]
INSERT(nums, 5, x)
REMOVE(nums, 1)
nums[4] ← nums[1]
FOR EACH item IN nums
{
  x ← x + item
}
DISPLAY(x)
```

ON SCRAP PAPER, create a trace table to help you determine what will be displayed when this algorithm is executed; you want to track all variables as they change.

What gets displayed?

Compare Solution & Video

ON SCRAP PAPER, create a trace table for the code **inside** the following procedure to help you see what value gets returned (***hint:*** trace `count` and `num`).

Here is a procedure called `mystery` that has parameters, `min` and `list`:

```
PROCEDURE mystery (min, list)
{
   count ← 0
   FOR EACH num IN list
   {
     IF (num > min)
     {
       count ← count + 1
     }
   }
   RETURN count
}
```

What gets displayed when these two lines of code get executed?

```
nums ← [6, 7, 2, 4, 8, 5, 10, -8]
DISPLAY(mystery(5, nums))
```

What gets displayed? i.e. what gets returned when `mystery(5, nums)` gets executed?	
Describe what the purpose of this procedure is:	

Compare Solution & See Video

DONE WITH PACKET 1

AP Practice Answer (back)

Correct answer is D

AP Exam Practice Question:
Which of the following statements about `PROCEDURE Mystery` are true?

```
PROCEDURE Mystery (number)
{
   REPEAT UNTIL (number ≥ 0)
   {
      number ← number + 2
   }
   IF (number = 0)
   {
      RETURN (true)
   }
   ELSE
   {
      RETURN (false)
   }
}
```

I. The procedure will return `true` for any negative, even integer.
II. The procedure will return `false` for any positive integer.
III. The procedure will return `false` for any odd integer, either positive and negative.

	(A) I only
	(B) II only
	(C) I and II only
X	(D) I, II, and III I is **true** because if you put in any negative, even integer, like -12, -10, -8, -6, etc. then the `repeat` loop continues adding 2 until `number` = 0, and then returns `true` (since `number` = 0). II is **true** because if you put in any positive integer, like 1, 2, 3, 4, 5, 6, etc. then the `repeat` loop doesn't iterate at all since `number` is already greater than or equal to 0! And since `number` does *not* equal 0, it reports `false`. III is **true** because if you put in an odd *negative* integer, like -7, -5, -3, -1, etc. the `repeat` loop iterates for negative numbers, adding 2 until `number` = 1. If you put in an odd, positive integer 1, 3, 5, 7, etc. then the `repeat` loop doesn't iterate at all since `number` is already greater than or equal to 0. Either way, `number` does *not* equal 0, so it reports `false`.

Answer (back)

The output is 15

Here's the value of each variable after each line of code:

Code	Values
`a ← 4`	**a = 4**
`b ← 5`	**b = 5**
`c ← 6 + a`	c = 6 + a = 6 + current value of a = 6 + 4 = 10 → **c = 10**
`a ← b`	a = b = 5 → **a = 5**
`c ← c + b`	c = c + b = current val of c + current val of b = 10 + 5 = 15
`DISPLAY(c)`	DISPLAY(15) → **15 is displayed**

Answer (back)

According to the Reference Sheet, the RANDOM command returns a random value between two values a and b (i.e. 4 and 10), ***including a and b.***

Text:
`RANDOM(a, b)`
Block:
`RANDOM` `a, b`
	Generates and returns a random integer from `a` to `b`, including `a` and `b`. Each result is equally likely to occur.
	
	For example, `RANDOM(1, 3)` could return `1`, `2`, or `3`.

So:

Here is a command written in pseudocode: `RANDOM(4, 10)`	a. What's the **smallest** number that could get randomly selected? **4** b. What's the **largest** number that ever gets randomly selected? **10**

Answer (back)

Take a look at this algorithm:
 	
`x ← 6`
`y ← 10`
`DISPLAY(RANDOM(x, y))`
	a. How many *different* numbers are possible to be displayed?
	
	5 different numbers are possible (*not* 4): 6, 7, 8, 9, and 10
	
	b. What's the likelihood (as a percentage) that the number displayed is a number greater than 8?
	
	Of the five possible numbers, the numbers 9 and 10 are greater than 8, so that's 2 out of 5 possible numbers, or 40%

Answer (back)

Here's some AP Exam pseudocode:
 	
`randomNum ← RANDOM(0, 9)`
`if (randomNum < 4)`
`{`
`  DISPLAY("Less than 4!")`
`}`
`else`
`{`
`  DISPLAY("Not less than 4!")`
`}`
	What is the likelihood (as a percent) that `"Not less than 4!"` is displayed?
	
	60%
	
	`randomNum` can be any of *ten* different numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9
	
	`"Not less than 4!"` gets displayed if randomNum is *not* less than 4, which are the numbers 4, 5, 6, 7, 8, and 9
	
	This is 6 out of 10, or 60%

Answer (back)

```
randomNum ← RANDOM(0, 9)
if (randomNum < 4)
{
  DISPLAY("Less than 4!")
}
else
{
  DISPLAY("Not less than 4!")
}
```

7. What is the likelihood (as a percent) that `"Not less than 4!"` is displayed?	**60%** randomNum can be any of *ten* different numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9 `"Not less than 4!"` gets displayed if randomNum is *not* less than 4, which are the numbers 4, 5, 6, 7, 8, and 9 This is 6 out of 10, or 60%

Answer (back)

Correct Answer: C

Multiple Choice

Peter wanted to write some code to choose a random number between 0 and 5, then print out whether the number was greater than 3, equal to 3, or less than three. Which of the following code segments would accomplish this?

Code Segment I

```
randomNum ← RANDOM(0, 5)
if (randomNum > 3)
{
  DISPLAY("Greater than 3!")
}
else
{
  if (randomNum = 3)
  {
    DISPLAY("Equal to 3!")
  }
  else
  {
    DISPLAY("Less than 3!")
  }
}
```

Code Segment II

```
randomNum ← RANDOM(0, 5)
if (randomNum > 3)
{
  DISPLAY("Greater than 3!")
}
if (randomNum = 3)
{
  DISPLAY("Equal to 3!")
}
if (randomNum < 3)
{
  DISPLAY("Less than 3!")
}
```

A. Code segment I only
B. Code segment II only
C. Both code segments
D. Neither code segment

Answer:	C

Answer (back)

Multiple choice: Which of the following will return an *even* random number between 1 and 10? A) `RANDOM(1, 5)` B) `RANDOM(2, 10)` C) `RANDOM(1, 10) / 2` **D) `RANDOM(1, 5) * 2`**	**D is correct** The `RANDOM(1, 5)` produces 1, 2, 3, 4, or 5, then multiplying that result by 2 will give 2, 4, 6, 8, or 10 -- an even number between 1 and 10! A is incorrect because it would return 1, 2, 3, 4 or 5 B is incorrect because it would return 2, 3, 4, 5, 6, 7, 8, 9, or 10 (not just even numbers) C is incorrect because it would FIRST pick a random number (1, 2, 3, 4, 5, 6, 7, 8, 9, or 10) THEN divide that number by 2, so it would return 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, or 5.

Answers (back)

Block-based:

```
PROCEDURE mystery [number]
    REPEAT UNTIL (number > 0)
        [number ← number + 4]
    IF (number = 2)
        [number ← number * 2]
    [RETURN number * 4]
```

Text-based:

```
PROCEDURE mystery (number)
{
   REPEAT UNTIL (number > 0)
   {
      number ← number + 4
   }
   IF (number = 2)
   {
      number ← number * 2
   }
   RETURN number * 4
}
```

Determine the value that gets returned by each of the following calls to the procedure:

Block-based:	**Text-based:**
mystery [-15]	Mystery (-15) = **4**
mystery [-10]	Mystery (-10) = **16**
mystery [-9]	Mystery (-9) = **12**
mystery [-8]	Mystery (-8) = **16**
mystery [-2]	Mystery (-2) = **16**
mystery [0]	Mystery (0) = **16**
mystery [2]	Mystery (2) = **16**
mystery [3]	Mystery (3) = **12**
mystery [5]	Mystery (5) = **20**
mystery [10]	Mystery (10) = **40**

Answers (back)

a.

```
i ← 0
sum ← 0
REPEAT UNTIL (i = 5)
    sum ← sum + i
    i ← i + 1
DISPLAY sum
```

What value will be displayed? **10**

Explanation:
First time through loop → sum = 0 i = 1
Second time through loop → sum = 1 i = 2
Third time through loop → sum = 3 i = 3
Fourth time through loop → sum = 6 i = 4
Fifth time through loop → **sum = 10 i = 5**

Loop ends after five iterations

b.

```
i ← 0
sum ← 0
REPEAT UNTIL (i > 5)
    i ← i + 1
    sum ← sum + i
DISPLAY sum
```

What value will be displayed? **21**

Explanation:
First time through loop → i = 1 sum = 1
Second time through loop → i = 2 sum = 3
Third time through loop → i = 3 sum = 6
Fourth time through loop → i = 4 sum = 10
Fifth time through loop → i = 5 sum = 15
Sixth time through loop → **i = 6 sum = 21**

Loop ends after six iterations

c.

```
i ← 0
sum ← 0
REPEAT UNTIL (i = 5)
    i ← 0
    sum ← sum + i
    i ← i + 1
DISPLAY sum
```

Why is this an **infinite loop**?
Because i *gets reset to 0 each time*:

```
i ← 0
sum ← 0
REPEAT UNTIL (i = 5)
    i ← 0
    sum ← sum + i
    i ← i + 1
DISPLAY sum
```

So even though 1 gets added, i will *always* be 1 after this step `i ← i + 1` so i will never be exactly 5 and so the loop will never terminate.

d.

```
i ← 0
sum ← 0
REPEAT UNTIL (i = 5)
    sum ← sum + i
    i ← i + 2
DISPLAY sum
```

This is *also* an infinite loop!
Because i *increments by 2 each time*:

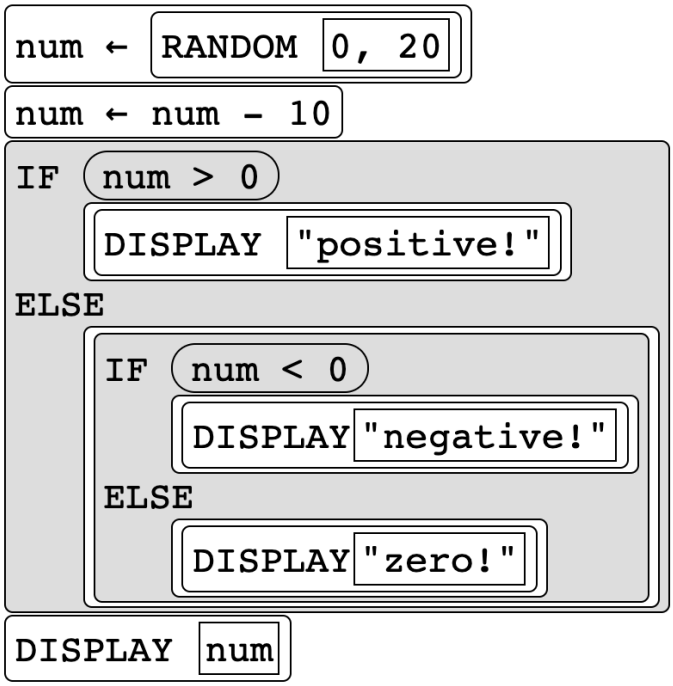
```
i ← 0
sum ← 0
REPEAT UNTIL (i = 5)
    sum ← sum + i
    i ← i + 2
DISPLAY sum
```

So i goes from 0 to 2 to 4 to 6 and ***skips over 5***, so i *will never be exactly 5* so the loop will never terminate.

Answer (back)

An algorithm is written below in both **text-based** and equivalent **block-based** pseudocode:

```
num ← RANDOM (0, 20)
num ← num - 10
IF (num > 0)
{
   DISPLAY ("positive!")
}
ELSE
{
   IF (num < 0)
   {
      DISPLAY ("negative!")
   }
   ELSE
   {
      DISPLAY ("zero!")
   }
}
DISPLAY (num)
```



What **two things** will be displayed if the random number selected in the first line is **6**?	If the `num` gets randomly set to is 6, then it gets set to `6 - 10 = -4`, so the *two* things that get displayed are "**`negative!`**" (since `num` is < 0) **and** "**`-4`**" at the very end (since the `DISPLAY (num)` comes after the if statements are done)
What **two things** will be displayed if the random number selected in the first line is **18**?	If the `num` gets randomly set to is 18, then it gets set to `18 - 10 = 8`, so the *two* things that get displayed are "**`positive!`**" (since `num` is > 0) **and** "**`8`**" at the very end:
What **two things** will be displayed if the random number selected in the first line is **10**?	If the `num` gets randomly set to is 18, then it gets set to `10 - 10 = 0`, so the *two* things that get displayed are "**`zero!`**" (since `num` is = 0) **and** "**`0`**" at the very end

Answer (back)

```
DISPLAY("Please enter a number:")
userNum ← INPUT() → user enters 20 here, so userNum = 20
result ← tripler(20) → 20 gets passed into tripler procedure, which
returns the value 60 (see below) → result = 60

DISPLAY("Your number tripled is: " + result) → Your tripled number is 60


-----------------------------
Calling the procedure:

PROCEDURE tripler(number)   → number = 20
{
  tripled ← number * 3      → tripled = 20 * 3 = 60
  RETURN tripled            → the value 60 gets returned
}
```

What will be displayed when the calling algorithm gets executed if the the user inputs the value "**20**" when prompted:	**Your tripled number is 60**

Answer (back)

Reminder! From the Reference Sheet, here is what LENGTH does:

Text:
`LENGTH(aList)`
Block:
`LENGTH aList`
	Evaluates to the number of elements in `aList`.

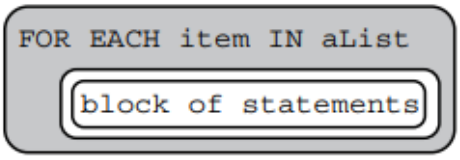
Here is some code written in AP Exam pseudocode that involves a **list**:

```
twoLetterWords ← ["ok", "it", "so", "pa", "no"]
num ← LENGTH(twoLetterWords) * 6
DISPLAY(num)
```

What will be displayed when this code is executed?	**30** The `LENGTH` is how many elements are in the list (and there are 5), and **5 * 6 = 30**

Answer (back)

Reminder! From the Reference Sheet, here is what FOR EACH does:

Text:
`FOR EACH item IN aList`
`{`
`  <block of statements>`
`}`
 	
Block:
 
	The variable `item` is assigned the value of each element of `aList` sequentially, in order, from the first element to the last element. The code in `block of statements` is executed once for each assignment of `item`.

Here is some code written in AP Exam pseudocode that involves a **list** and a **for each** loop:

```
numberList ← [10, -3, 5, 0, -7, 4, 8, -6]
FOR EACH item IN numberList
{
  IF (item > 0)
  {
    DISPLAY(item)
  }
}
```

What will be displayed when this code is executed?	**10** **5** **4** **8** **Reminder!** In a **for each loop,** *every* element gets *checked*, and `item` takes on the value of each number, *in order* (10, then -3, then 5, then 0, then -7, then 4, then 8, then -6). The if statement then decides which gets *displayed,* **but *all* 8 numbers in the list are checked!**

AP Answer (back)

Correct answer is B

AP EXAM PRACTICE QUESTION

A programmer wrote the following program to determine the **average** of a list of numbers called `numberList`:

```
numberList ← 1, 2, 3, 4
sum ← 0
FOR EACH number IN numberList
    sum ← sum + number
DISPLAY sum / LENGTH numberList
```

If the program displays 2.5, which statement is **true**?

X the correct answer choice.

	A. The result is correct, and the single test case with numberList = [1, 2, 3, 4] is sufficient for concluding that the program will work for all possible lists of numbers
	B. The result is correct, but the single test case where numberList = [1, 2, 3, 4] is not sufficient for concluding that the program will work for all other lists of numbers, and the programmer should test additional numberLists. **EXPLANATION** The average of 1, 2, 3, and 4 is 2.5 so it's correct, BUT using only one test case is never sufficient for ensuring the program works for ALL possible number lists! Programmers should ALWAYS test lots of possible inputs! Note that choice C isn't true because the program, as written, would work for lists other than length 4 (since it uses a FOR EACH loop).
	C. The result is correct, but the program only works for numberLists that contain exactly four elements.
	D. The result is incorrect, so the program has an error in it that the programmer should find and debug.

AP Answer (back)

Correct answer is B

AP EXAM PRACTICE QUESTION

A programmer wrote the following procedure to determine the average of a list of numbers called `numberList`:

```
Line 1:  PROCEDURE CalculateAverage(numberList)
Line 2:  {
Line 3:     sum ← 0
Line 4:     count ← 1
Line 5:     FOR EACH number IN numberList
Line 6:     {
Line 7:        sum ← sum + number
Line 8:        count ← count + 1
Line 9:     }
Line 10:    average ← sum / count
Line 11:    DISPLAY (average)
Line 12: }
```

This procedure will not correctly calculate the average; which change to the code will fix the error?

X the correct answer choice.

	A. Change line 3 to `sum ← numberList[1]`
	B. Change line 4 to count ← 0 **Explanation** Changing line 4 to count ← 0 fixes the issue since "count" is being used to determine how many elements are in the list (so it can divide by that many), but by starting it at 1 (rather than 0), it will be 1 greater than the length of the list, so the average won't calculate correctly. The other answers are incorrect because the issue is that count shouldn't be starting at 1, it should be starting at 0.
	C. Interchange line 7 and line 8
	D. Change line 11 to `DISPLAY (average - 1)`

Practice AP Question (back)

Correct answer is C

18. Practice AP Question!

Which of the following algorithms output the value 3?

Algorithm I

```
grades ← []
APPEND(grades, 2)
APPEND(grades, 3)
APPEND(grades, 1)
DISPLAY(grades[2])
```

Algorithm II

```
numbers ← [3, 1]
APPEND(numbers, 2)
sum ← numbers[2] + numbers[3]
DISPLAY(sum)
```

A. Algorithm I only
B. Algorithm II only
C. Both Algorithms I and II
D. Neither Algorithm I nor II

Explanation

Algorithm I
Here is what happens after each line of code:

```
grades ← []            → grades = []
APPEND(grades, 2)      → grades = [2]
APPEND(grades, 3)      → grades = [2, 3]
APPEND(grades, 1)      → grades = [2, 3, 1]
DISPLAY(grades[2])     → grades[2] = 3, which gets displayed
```

Algorithm II
Here is what happens after each line of code:

```
numbers ← [3, 1]                  → numbers = [3, 1]
APPEND(numbers, 2)                → numbers = [3, 1, 2]
sum ← numbers[2] + numbers[3]     → numbers[2] = 1 and numbers[3] = 2
                                    → so sum = 1 + 2 = 3
DISPLAY(sum)                      - sum is 3, which gets displayed
```

Answer (back)

Correct answer is C

WARM UP QUESTION

Below is a variable `animals` that stores a list:

```
animals ← ["cat", "fish", "lizard", "giraffe", "dog"]
```

What is the value of `animals[4]`?

X the correct answer choice.

	A. `"m"`
	B. `"fish"`
	C. **`"giraffe"`** **Explanation** `animals[4]` means the 4th element in the animals list, which is "giraffe"
	D. `5`

Answer (back)

Reminder! From the Reference Sheet, here is the relevant "assignment" command for this problem:

Text:
`x ← aList[i]`
Block:
`x ← aList [i]`
	Assigns the value of `aList[i]` to the variable `x`.

Here is some code written in AP Exam pseudocode that involves a **list** and a **for each** loop:

```
numberList ← [5, 7, 4, 10, 6, 2]
num1 ← numberList[2]
DISPLAY(num1)
DISPLAY(numberList[4])
```

What will be displayed when this code is executed?	Two numbers get displayed: **7 (the second element in numberList)** **10 (the fourth element in numberList)**

Answer (back)

Correct answer is A

WARM UP QUESTION

Below is a variable `animals` that stores a list:

```
animals ← ["cat", "fish", "lizard", "giraffe", "dog"]
```

What is the value of `LENGTH(animals)`?

X the correct answer choice.

	A. `5` **Explanation** `LENGTH(animals)` is taking the **length** of a list, which returns the number of elements in the list; there are 5 elements, so the length is 5.
	B. `7`
	C. `[3, 4, 6, 7, 3]`
	D. `None of the above`

Answer (back)

Correct answer is B

AP EXAM PRACTICE QUESTION

Selena wrote the following code:

```
m ← 2
n ← 1
subjects ← ["math", "ELA", "AP CSP", "science"]
mystery ← subjects[n]
REPEAT m TIMES
{
    n ← n + 1
    mystery ← subjects[n]
}
DISPLAY(mystery)
```

What value is displayed?

X the correct answer choice.

	A. `"math"`
	B. `"AP CSP"` **Explanation** In this code, the REPEAT iterates 2 times (m = 2), and each time thru the loop, n increases by 1, THEN mystery gets updated to subjects[n]. The first time through the loop, n increases to 2, then mystery gets updated to subjects[2], which is "ELA" During the 2nd (and last) iteration of the REPEAT loop, n increases to 3, and mystery is set to subjects[3], which is "AP CSP"
	C. `"science"`
	D. `["math", "ELA", "AP CSP"]`

Answer (back)

Correct answer is C

Practice AP Question

Which of the following algorithms output the value 29?

Algorithm I

```
temps ← [31, 30, 29]
REMOVE(temps, 2)
DISPLAY(temps[2])
```

Algorithm II

```
numbers ← []
APPEND(numbers, 30)
APPEND(numbers, 29)
REMOVE(numbers, 2)
value ← numbers[1] - LENGTH(numbers)
DISPLAY(value)
```

A. Algorithm I only
B. Algorithm II only
C. Both Algorithms I and II
D. Neither Algorithm I nor II

Explanation

Algorithm I
Here is what happens after each line of code:

```
temps ← [31, 30, 29]    → temps = [31, 30, 29]
REMOVE(temps, 2)        → temps = [31, 29]
DISPLAY(temps[2])       → temps[2] = 29, which gets displayed
```

Algorithm II
Here is what happens after each line of code:

```
numbers ← []                            → numbers = []
APPEND(numbers, 30)                     → numbers = [30]
APPEND(numbers, 29)                     → numbers = [30, 29]
REMOVE(numbers, 2)                      → numbers = [30]
value ← numbers[1] - LENGTH(numbers) → numbers[1] = 30 and
                                          LENGTH(numbers) = 1
                                          (since there is only one element remaining)
DISPLAY(value)                       → value = 30 - 1 = 29, which
                                          gets displayed
```

Answer (back)

Correct answer is C

AP EXAM PRACTICE QUESTION

Here is some code that executes:

```
num ← 4
mysteryList ← [num, 3]
APPEND(mysteryList, 7)
INSERT(mysteryList, 1, 2)
INSERT(mysteryList, 1, num)
INSERT(mysteryList, num, 5)
DISPLAY(mysteryList)
```

Which list accurately reflects the `mysteryList` that gets displayed at the end?

A. `[4, 4, 2, 3, 5, 7]`
B. `[4, 1, 3, 4, 4, 7]`
C. **`[4, 2, 4, 5, 3, 7]`**
D. None of the above

EXPLANATION

Code	**Value of `mysteryList` after each line of code:**	
`num ← 4`		
`mysteryList ← [num, 3]`	`[4, 3]`	
`APPEND(mysteryList, 7)`	`[4, 3, 7]`	
`INSERT(mysteryList, 1, 2)`	`[2, 4, 3, 7]`	insert 2 at index 1
`INSERT(mysteryList, 1, num)`	`[4, 2, 4, 3, 7]`	insert 4 at index 1
`INSERT(mysteryList, num, 5)`	`[4, 2, 4, 5, 3, 7]`	insert 5 at index 4
`DISPLAY(mysteryList)`	**`[4, 2, 4, 5, 3, 7]`**	

Answer (back)

Correct answer is B

Practice AP Question

Here is some code that executes:

```
cakeIngredients ← ["butter", "flour", "eggs", "oil"]
cakeIngredients[2] ← "sugar"
cakeIngredients[4] ← cakeIngredients[3]
DISPLAY(cakeIngredients)
```

Which list accurately reflects the `mysteryList` that gets displayed at the end?

A. `["butter", "flour", "eggs", "eggs"]`

B. **`["butter", "sugar", "eggs", "eggs"]`**

C. `["butter", "sugar", "oil", "oil"]`

D. `["butter", "sugar", "eggs", "oil"]`

Explanation	**cakeIngredients after each line of code**
`cakeIngredients ← ["butter", "flour", "eggs", "oil"]`	→ `["butter", "flour", "eggs", "oil"]`
`cakeIngredients[2] ← "sugar"`	→ `["butter", "sugar", "eggs", "oil"]`
`cakeIngredients[4] ← cakeIngredients[3]`	→ `["butter", "sugar", "eggs", "eggs"]`
`DISPLAY(cakeIngredients)`	→ **`["butter", "sugar", "eggs", "eggs"]`**

Answer (back)

Here is an algorithm:

```
a ← 3
b ← 1
REPEAT 5 TIMES
{
  a ← a + b
  b ← b + 1
}
DISPLAY(a + b)
```

a. Complete the trace table to help you determine what will be displayed at the end.

a	b	Iteration #
3	1	
3 + 1 = 4	1 + 1 = 2	1
4 + 2 = 6	2 + 1 = 3	2
6 + 3 = 9	3 + 1 = 4	3
9 + 4 = 13	4 + 1 = 5	4
13 + 5 = 18	5 + 1 = 6	5

The final value of a after iteration number 5 is **18** and the final value of b is **6**

b. What will be displayed?

18 + 6 = 24 → **24 gets displayed**

Answer (back)

2. Here is an algorithm written in AP exam pseudocode:

```
numList ← [8, 7, 9, 5]
len ← LENGTH(numList)
count ← 1
sum ← 0

REPEAT len TIMES
{
  sum ← sum + numList[count]
  count ← count + 1
}

DISPLAY(sum + count)
```

Here is a trace table you might create; in this problem, only `count` and `sum` change

numList	len	count	sum	iteration #
[8, 7, 9, 5]	4	1	0	
		2	8	1
		3	15	2
		4	24	3
		5	**29**	4

What gets displayed?

34 (29 + 5)

If you didn't quite get this, here is a solution video with Mr. Miller explaining it.

Answer (back)

Here is an algorithm written in AP exam pseudocode:

```
num ← 10
output ← 5
REPEAT UNTIL (num < 0)
    IF (num > output)
        output ← output + num
    ELSE
        output ← output - num
    num ← num - 2
DISPLAY output
```

Complete the trace table to help you determine what will be displayed when this algorithm is executed:

num	output
10	5
8	15
6	7
4	1
2	5
0	3
-2	**3**

What gets displayed?

3

If you didn't quite get this, here is a solution video with Mr. Miller explaining it.

Answer (back)

Here is an algorithm in pseudocode:

```
nums ← [2, 5, 7]        →
INSERT(nums, 2, 9)      →
APPEND(nums, 3)         →
APPEND(nums, 8)         →
APPEND(nums, 4)         →
nums[3] ← 1             →
REMOVE(nums, 1)         →
INSERT(nums, 5, 6)      →
DISPLAY(nums)           →
```

Complete the trace table (started for you):

nums
[2, 5, 7]
[2, **9**, 5, 7]
[2, 9, 5, 7, **3**]
[2, 9, 5, 7, 3, **8**]
[2, 9, 5, 7, 3, 8, **4**]
[2, 9, **1**, 7, 3, 8, 4]
[9, 1, 7, 3, 8, 4] **(2 removed)**
[9, 1, 7, 3, **6**, 8, 4]

What gets displayed?

[9, 1, 7, 3, 6, 8, 4]

Answer (back)

What gets displayed when the following code segment is executed?

```
nums ← [3, 1, 6, 4]
INSERT(nums, 3, 5)
APPEND(nums, 2)
nums[4] ← 1
REMOVE(nums, 2)
x ← nums[3]
INSERT(nums, 5, x)
REMOVE(nums, 1)
nums[4] ← nums[1]
FOR EACH item IN nums
{
  x ← x + item
}
DISPLAY(x)
```

Here is a trace table you might have made (see this video if you want to see how Mr. Miller created it):

nums	x	Item
[3, 1, 6, 4]	1	5
[3, 1, 5, 6, 4]	6	1
[3, 1, 5, 6, 4, 2]	7	4
[3, 1, 5, 1, 4, 2]	11	5
[3, 5, 1, 4, 2]	16	2
[3, 5, 1, 4, 1, 2]	18	
[5, 1, 4, 1, 2]		
[5, 1, 4, 5, 2]		

What gets displayed?

18

If you didn't quite get this, be sure to check out the solution video!

Answer (back)

Here is a procedure called `mystery` that has parameters, `min` and `list`:

```
PROCEDURE mystery (min, list)
{
   count ← 0
   FOR EACH num IN list
   {
     IF (num > min)
     {
       count ← count + 1
     }
   }
   RETURN count
}
```

What gets displayed when these two lines of code get executed?

```
nums ← [6, 7, 2, 4, 8, 5, 10, -8]
DISPLAY(mystery(5, nums))
```

Here is a trace table you might have made (see this video if you want to see how Mr. Miller created it):

count	num
0	6
1	7
2	2
3	4
4	8
	5
	10
	-8

What gets displayed? i.e. what gets returned when `mystery(5, nums)` gets executed?

4

Describe what the purpose of this procedure is:

Returns the number of values in the list that are greater than min

If you didn't quite get this, be sure to check out the solution video!

Check (back)

Here is a procedure named `mystery` that has a list of numbers as a parameter:

```
PROCEDURE mystery(num_list)
{
  count = 0
  FOR EACH num IN num_list
  {
    IF num > 5
    {
      DISPLAY(num)
      count ← count + 1
    }
  }
  DISPLAY("end!")
  RETURN count
}
```

What gets displayed when the following code is executed?

```
result ← mystery([1, 6, 2, 7, 3, 8, 4, 9, 5])    # result is 4
DISPLAY("the result is: " + result)
```

What gets displayed?	
	6 **7** **8** **9** **end!** **the result is: 4** **EXPLANATION:** The numbers in the list in purple get displayed by the blue display statement, then the pink display statement prints after the loop has checked all values in the list, then lastly 4 gets returned and displayed by the orange display statement in the main program: **6** **7** **8** **9** **end!** **the result is: 4**

Check (back)

When a return statement is reached in a procedure, the procedure ends immediately and the value gets returned. No more code inside the procedure executes. If the return happens inside a loop, the loop also ends.

Here is a similar procedure named `mystery` that has a list of numbers as a parameter:

```
PROCEDURE mystery(num_list)
{
  count = 0
  FOR EACH num IN num_list
  {
    IF num > 5
    {
      DISPLAY(num)
      count ← count + 1
      RETURN count
    }
  }
  DISPLAY("end!")   DOES NOT GET EXECUTED
  RETURN count      DOES NOT GET EXECUTED
}
```

What gets displayed when the following code is executed?

```
result ← mystery([1, 6, 2, 7, 3, 8, 4, 9, 5]) # result is 1
DISPLAY("the result is: " + result)
```

What gets displayed?	**6** **the result is: 1** **EXPLANATION:** Only the 6 gets displayed by the **blue** display statement because the **return** statement gets reached after checking the 6, ***which ends the loop (no more numbers in the list are checked) and no more code in the procedure executes.*** The value of 1 gets returned, which then gets displayed by the orange display statement: **6** **the result is: 1**