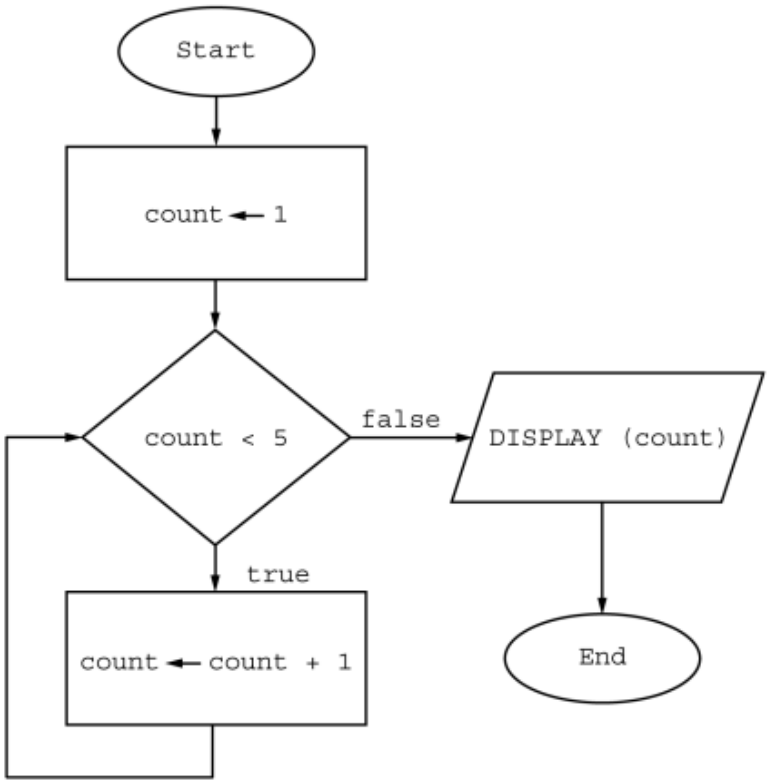
Set 31: While loops

Skill 31.01: Interpret loops in program flow charts

Skill 31.01 Concepts

A loop is a data structure that allows us to repeat a block of code until a specified condition is met. The flow chart below illustrates the use of a loop to increment the variable *count*. Notice that in the example, *count* gets incremented until it reaches the value of 5, after which time the loop is exited and the result is displayed.



What is displayed as a result of executing the algorithm in the flowchart?

(A) 5

(B) 15

(C) 1 2 3 4

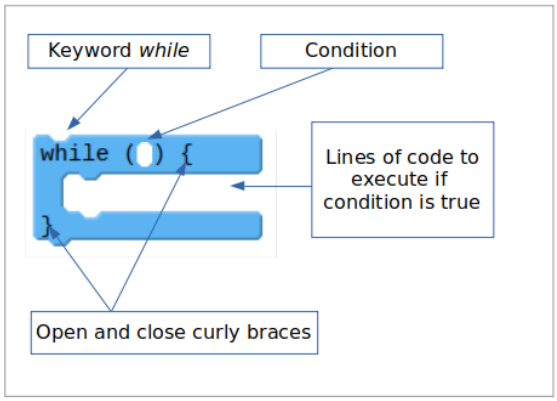
(D) 1 2 3 4 5

Skill 31.01 Exercise

Skill 31.02: Interpret While loop pesudocode

Skill 31.02 Concepts

The *while* loop uses a *boolean* condition to repeatedly run a block of code. It checks the expression, and if it is true it runs the block of code contained within it. This process of checking the condition and running the block of code is repeated as long as the *boolean* condition remains true. **Once the boolean expression becomes false it will stop. To the right is a diagram showing the elements of a basic *while loop*.**



Before we get started writing *while loops*, lets practice some pseudocode scenarios. Although the examples below do use the while key word, they can still be interpreted the same way. That is, if a condition is true, keep looping!

Example

```
row = 0;

WHILE(row <= 4){
    MOVE_TO[row][col]
    FILL(grey)
    row = row + 1
}
```

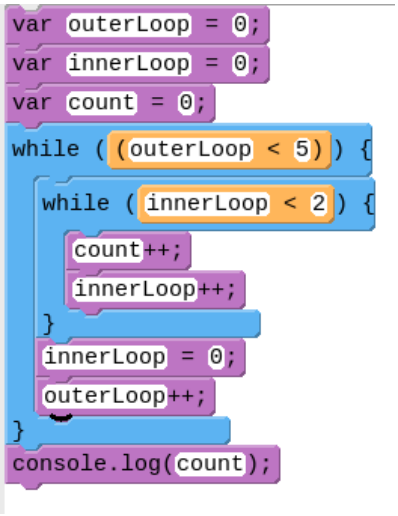
		COLS				
		0	1	2	3	4
ROWS	0					
	1					
	2					
	3					
	4					

[Skill 31.02 Exercises 1 thru 4]

Skill 31.03 Interpret nested while loop pseudocode

Skill 31.03 Concepts

While loops can also appear inside a while loop! In the example to the right, the outer loop repeats 5 times and the inner loop repeats 2 times for a total of 5 x 2 = 10 times! This is because the outer loop cannot continue until the innerLoop has completed executing. So, when the outerLoop is 0, the innerLoop executes 2 times, when the outerLoop is 1, the innerLoop executes 2 times, so on and so forth, so the total count equals 2 x 5, or 10 at the end.



Skill 31.03 Exercise 1

Example

```
row = 0;
col = 0;

WHILE(row <= 4){
    WHILE(col <=4){
        MOVE_TO[row][col]
        if((col MOD 2)EQUALS(0)){
            FILL(grey)
        }
        col = col + 1;
    }
col = 0
row = row + 1
}
```

		COLS				
		0	1	2	3	4
ROWS	0					
	1					
	2					
	3					
	4					

Skill 31.03 Exercise 2

Skill 31.04 Identify an infinite while loop

Skill 31.04 Concepts

while loops run until their condition becomes false, which raises an interesting question. **What happens if the condition never becomes false**? In these cases the program enters what is called an i**nfinite loop** over the commands in the *while loop*, and it never reaches the rest of your program. In the code below, what if the line indicated in the box was instead changed to `number ← number + 2`? If this were true, the procedure would never end, in fact your program would crash! For this reason, w**e normally avoid infinite loops in our programs.**

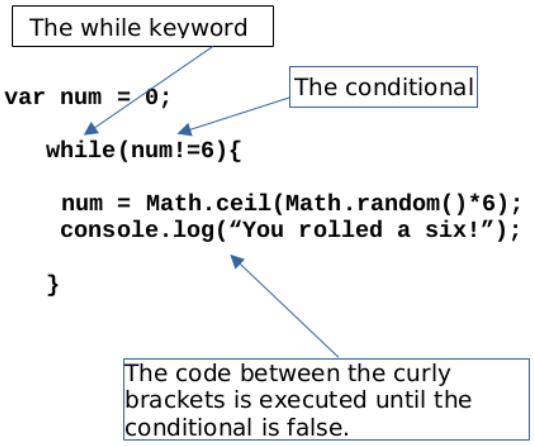
```
PROCEDURE Mystery (number)
{
    REPEAT UNTIL (number ≤ 0)
    {
        number ← number - 2
    }
    IF (number = 0)
    {
        RETURN (true)
    }
    ELSE
    {
        RETURN (false)
    }
}
```

[Skill 31.04 Exercises 1 & 2]

Skill 31.05 Write a while loop in JavaScript

Skill 31.05 Concepts

The syntax for writing a while loop in javascript is shown to the right. Just as we saw with the pseudocode examples, the *while loop* **uses a *boolean* condition to repeatedly run a block of code.** It checks the expression, and if it is true it runs the block of code contained within it. This process of checking the condition and running the block of code is repeated as long as the *boolean* condition remains true. **Once the boolean expression becomes false it will stop.**



[Skill 31.05 Exercises 1 & 2]