

Logo Activity 12 Recursion With Incrementing

Have you ever wanted to write a program that counted something? Perhaps you wanted the user to say how many times they wanted a message to appear. You can write programs that counts by building loops using recursion. Try this procedure:

```
TO COUNT.UP
  :INPUT.VALUE
  PRINT
  :INPUT.VALUE
  WAIT 10
  COUNT.UP :INPUT.VALUE + 1
END
```

Type COUNT.UP 1Logo responds with:

```
1
2
3
4
forever!
```

There is no stop rule. The only way to stop the process is to press the HALT button. Let's examine what is happening. Begin with a value of 1 for the input.

```
TO COUNT.UP 1
  PRINT 1
  WAIT 10
  COUNT.UP 1 + 1
END
```

```
    TO COUNT.UP 2
      PRINT 2
      WAIT 10
      COUNT.UP 2 + 1
    END
```

```
        TO COUNT.UP 3
          PRINT 3
          WAIT 10
          COUNT.UP 3 + 1
        END
```

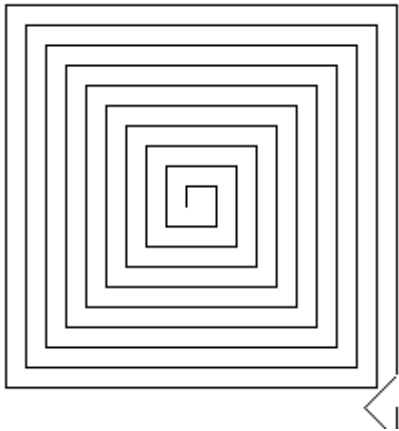
To stop the procedure at 10, use an IF statement as follows:

```
TO COUNT.UP :INPUT.V ALUE
  PRINT :INPUT.VALUE
  WAIT 10
  IF ( :INPUT.VALUE < 10 ) [COUNT.UP :INPUT.VALUE + 1]
END
```

Using Incrementing

Incrementing can be used in a variety of settings. Suppose you want to make a square spiral out, that is, a square in which each side gets longer as each side is drawn. You need to add a fixed amount each time you draw a side. This is similar to the COUNT.UP program above. You could write:

```
TO GROWSQUARE :SIDE
  FD :SIDE
  RT 90
  IF (:SIDE <200) [GROWSQUARE :SIDE + 5]
END
```



If you type GROWSQUARE 10, the turtle draws a side of length 10, then a side of length 15, then a side of 20, and so forth until the side reaches 200.

Recursion in Reverse

Bet you can't predict this one!

```

TO COUNT.UP.DOWN :INPUT.VALUE
  PRINT :INPUT.VALUE
  WAIT 5
  IF (:INPUT.VALUE < 5)[COUNT.UP.DOWN :INPUT.VALUE + 1]
  WAIT 5
  PRINT :INPUT.VALUE
END
  
```

The out put appears as follows:

```

1
2
3
4
5           ← Last value pushed onto the stack
5           ← First value popped off the stack
4
3
2
1
  
```

The first part of the program is easy to figure out. It just prints each value as it places it on a stack. The statements that follow the IF are only executed after the base case (:INPUT.VALUE < 5) is no longer true. So, what does this stack look like:

Pushing onto the stack Popping values off the stack

				5
			4	4
		3	3	3
	2	2	2	2
1	1	1	1	1

5				
4	4			
3	3	3		
2	2	2	2	
1	1	1	1	1

Bold faced values are those that are output (as shown above).

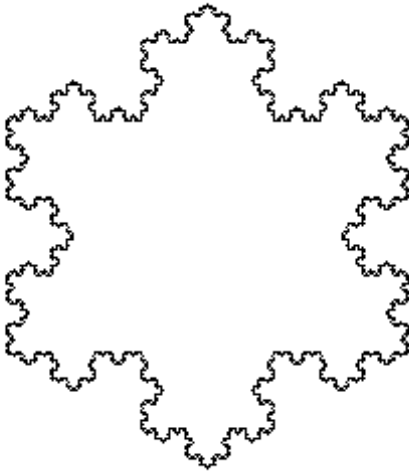
This type of recursion can have some unexpected results. It can produce some very elegant procedures with some beautiful results. The growing snowflake is one such example.

The Growing Snowflake

This code is just amazing:

```
to koch :x
  repeat 3 [triline :x rt 120]
end
to triline :x
  if :x < 1 [fd :x] [triline :x/3 lt 60 triline :x/3 rt 120 triline
:x/3 lt 60 triline :x/3]
end
```

Test it with
RT 30
KOCH 200



While this code is far beyond the scope of this class, it is a great example of a fractal image that Logo is capable of producing with ease. Students trying to program this same image in other programming languages may take pages of code to produce the same results.

Logo Assignment 12 (10pts) Name _____

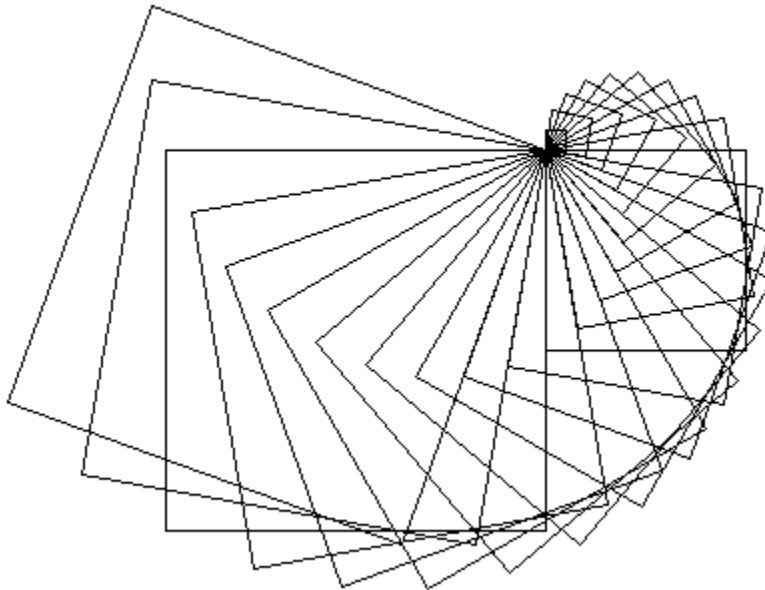
- 1) Create a COUNT. DOWN program that will count down from 10 to 1 by 1's. Report your results here.

- 2) In a previous lesson you created a program that required a user to enter a password in order to see a design. Modify this program by allowing the user only three chances at entering the correct password. If the correct password is entered the design will appear. Otherwise, a message about an intruder will be displayed and the program will end. Report your procedures here.

- 3) In this problem, you will experiment with grow procedures similar to the square spiral in this lesson. Your task will be to make a triangle or hexagonal spiral.

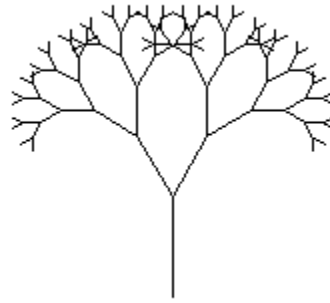
- 4) Many recursive procedures make designs that look very artistic. You are to create your own design by following the steps below. These steps will help you create a recursive procedure that will make an object grow and spin a finite number of times. Create a spinning square, triangle, pentagon, flag, club, or another object.
- Think of a shape for the turtle to draw.
 - Begin writing a procedure called `SPIN.N.GROW :SIZE`
 - Type the commands to draw your basic shape.
 - Make a small turn to the right or left.
 - Stop the recursive loop if `SIZE` exceeds a maximum value.
 - Type the name of the procedure on the last line of the procedure.

Report your results here!



- 5) Type in the following code. It will create a Logo tree. The tree will be equally distributed on both sides. In other words, all of the branches will be the same length.

```
to tree :size
  if (:size > 5) [fd :size lt
30 tree :size * 0.7 rt 60 tree
:size * 0.7 lt 30 bk :size]
end
```



Your task will be to create a tree with random length branches. Place your final code below.