Recursion: advantages

Challenge 2

A recursive algorithm can always be written to use iteration (instead of recursion). The following statements show possible advantages of using recursion. Which **two** are correct?

- [ ] Recursive algorithms use less memory
- [ ] Problems that are naturally expressed by recursion are easier to code
- [ ] There are usually fewer lines of code
- [ ] Recursive algorithms are easier to trace

Recursion: identify base case

Practice 2

A recursive subroutine has been written as follows:

Pseudocode

```
FUNCTION power(n, exp)
    IF exp == 0 THEN
        RETURN 1
    ELSE
        RETURN n * power(n, exp-1)
    ENDIF
ENDFUNCTION
```

What is the base case in this subroutine?

Quiz:
STEM SMART Computer Science Week 9

Recursion: trace table

Challenge 2

A recursive subroutine has been written below. A trace table has been partially filled in, showing the state of the variables and condition, alongside the number of calls to the `make_num` subroutine and the return value.

Write the correct value in the correct location in the trace table:

Pseudocode

```
FUNCTION make_num(x, y)
    IF x == 0 THEN
        RETURN y
    ELSE
        RETURN make_num(x-2, y-1)
    ENDIF
ENDFUNCTION

make_num(6, 7)
```

Call #	x	y	x equals 0?	Return
1	6	7	False	make_num(4, 6)
2				
3				
4				

Quiz:
STEM SMART Computer Science Week 9

Recursion: trace the code 1

Challenge 2

A recursive subroutine has been written as follows:

Pseudocode

```
FUNCTION do_something(x, y)
    IF x == 1 THEN
        RETURN y
    ELSE
        RETURN do_something(x-1, x+y)
    ENDIF
ENDFUNCTION
```

Trace the subroutine to determine what the final return value will be when the following call is made:

`do_something(5, 2)`

Recursion: trace the code 2

Challenge 2

A recursive subroutine has been written as follows:

Pseudocode

```
FUNCTION do_something(x, y)
    IF x == 1 THEN
        RETURN y
    ELSE IF y == 1 THEN
        RETURN x
    ELSE
        RETURN do_something(x-1, y-2)
    ENDIF
ENDFUNCTION
```

Trace the subroutine to determine what the final return value will be when the following call is made:

`do_something(4, 8)`

Quiz:
STEM SMART Computer Science Week 9

Recursion: trace the code 3

Practice 2

A recursive subroutine has been written as follows:

Pseudocode

```
FUNCTION do_something(n)
    IF n == 1 THEN
        RETURN 0
    ELSE
        RETURN 1 + do_something(n DIV 2)
    ENDIF
ENDFUNCTION
```

`DIV` performs an integer division. For example, `42 DIV 10` will return 4 since this is the whole number of times that 10 goes into 42.

When the subroutine is run with the value 18 specified as the argument, i.e. `do_something(18)`, the subroutine returns the value 4.

What value is returned when the subroutine is run with the value 100 specified as an argument, i.e. `do_something(100)`?

Quiz:
STEM SMART Computer Science Week 9

Recursion: trace the code 4

Challenge 2

Amir wrote a recursive subroutine that outputs a pattern resembling festive garlands. It does that by using the star symbol *.

For example, `garland(3)` produces the following output:

```
***
**
*
*
**
***
```

Pseudocode

```
PROCEDURE garland(x)
    PRINT('*' * x)
    IF x > 1 THEN
        garland(x-1)
    ENDIF
    PRINT('*' * x)
ENDPROCEDURE
```

Trace the subroutine to determine **how many times the subroutine `garland` is called in total** when it is run with the value 5 as an argument:

`garland(5)`

Quiz:
STEM SMART Computer Science Week 9

Recursion: complete missing code

Practice 2

Fibonacci numbers are a number sequence that starts: 0, 1, 1, 2, 3, 5, 8, 13, ...

Each Fibonacci number is the **sum of the previous two numbers**. The table below shows the first 10 Fibonacci numbers, from F_0 to F_9:

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
0	1	1	2	3	5	8	13	21	34

An incomplete recursive function `fibonacci(n)` has been written below to output the n th Fibonacci number, F_n. For example, running the function with the argument value 7 would return 13 (as the Fibonacci number F_7 is 13).

Hint 2 contains a breakdown of the rules for the Fibonacci sequence if you need help with completing the missing code.

Pseudocode

```
FUNCTION fibonacci(n)
    IF n == 0 THEN
        RETURN [a]
    ELSEIF n == 1 THEN
        RETURN [b]
    ELSE
        RETURN [c]
    ENDIF
ENDFUNCTION
```

Part A Code for [a]

What should replace **[a]**?

Part B Code for [b]

What should replace **[b]**?

Part C Code for [c]

What should replace **[c]**?

Quiz:
STEM SMART Computer Science Week 9

Recursion: purpose of subroutine 1

Challenge 2

A recursive subroutine has been written as follows:

Pseudocode

```
FUNCTION do_something(n)
    IF n != 0 THEN
        do_something(n DIV 2)
        PRINT(n MOD 2)
    ENDIF
ENDFUNCTION
```

The subroutine must be passed a whole number as an argument (e.g. 57).

What problem does this subroutine solve?

- Calculates the square root of a number
- Converts a denary number into binary
- Converts a positive number to negative (2's complement)
- Rounds the number to 2 decimal places