Recursion: advantages

A Level
C C

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?

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

UNIVERSITY OF CAMBRIDGE Raspberry Pi Foundation

All teaching materials on this site are available under a CC BY-NC-SA 4.0 license, except where otherwise stated.

Recursion: identify base case

A Level
P P

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?

UNIVERSITY OF CAMBRIDGE Raspberry Pi Foundation

All teaching materials on this site are available under a CC BY-NC-SA 4.0 license, except where otherwise stated.

Recursion: complete missing code

A Level
P P

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**.

A recursive subroutine has been written to output the nth Fibonacci number.
For example, running the subroutine with the argument value 7, would return 13
(as 13 is the 7th Fibonacci number).

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]**?

UNIVERSITY OF CAMBRIDGE Raspberry Pi Foundation

All teaching materials on this site are available under a CC BY-NC-SA 4.0 license, except where otherwise stated.

Recursion: trace code 1

A Level
C C

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)`

UNIVERSITY OF CAMBRIDGE Raspberry Pi Foundation

All teaching materials on this site are available under a CC BY-NC-SA 4.0 license, except where otherwise stated.

Recursion: purpose of subroutine 1

A Level
C C

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?

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

UNIVERSITY OF CAMBRIDGE Raspberry Pi Foundation

All teaching materials on this site are available under a CC BY-NC-SA 4.0 license, except where otherwise stated.

Recursion: trace code 4

A Level
C C

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 will call itself** when it is run with the value 5 as an argument:

garland(5)

UNIVERSITY OF CAMBRIDGE   Raspberry Pi Foundation

All teaching materials on this site are available under a CC BY-NC-SA 4.0 license, except where otherwise stated.

Recursion: trace code 2

A Level

P P

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
```

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)`?

UNIVERSITY OF CAMBRIDGE   Raspberry Pi Foundation

All teaching materials on this site are available under a CC BY-NC-SA 4.0 license, except where otherwise stated.

Recursion: passing values

A Level
C C

A recursive subroutine has been written as follows:

Pseudocode

```
FUNCTION sum_to_n(n:BYVAL)
    IF n == 1 THEN
        RETURN 1
    ELSE
        RETURN n + sum_to_n(n-1)
    ENDIF
ENDFUNCTION
```

Why must the argument be passed by *value* (and not by *reference*)?

- ○ If a shared reference is used, you will encounter 'stack overflow'
- ○ If a shared reference is used, every recursive call will use the same value for n
- ○ A value is used so that the subroutine knows that n is a number
- ○ A value is used so that the calculation does not throw a runtime error

UNIVERSITY OF CAMBRIDGE   Raspberry Pi Foundation

All teaching materials on this site are available under a CC BY-NC-SA 4.0 license, except where otherwise stated.

Recursion: trace code 3

A Level
C C

Scores is an array that contains the test scores of each student from a recent class test. Each element of the array contains a record containing the student’s surname, first name, and score. These values are accessed using the syntax:

```
student.surname
student.forename
student.score
```

The array is ordered by surname and contains records for these students:

Ahmed, Atkinson, Bashir, Bell, Chesworth, Cooper, Endover, Faujdar, James, Khan, Mohammad, Murray, Singh, Williams, Young

The teacher wants to know what score was achieved in the test by the student with the surname James. The search is carried out using the recursive subroutine defined below.

Pseudocode

```
PROCEDURE search(surname, scores, first, last)
    IF first > last THEN
        PRINT("Not found")
    ELSE
        mid = (first + last) DIV 2
        IF scores[mid].surname == surname THEN
            PRINT(scores[mid].score)
        ELSEIF surname > scores[mid].surname THEN
            first = mid + 1
            search(surname, scores, first, last)
        ELSE
            last = mid - 1
            search(surname, scores, first, last)
        ENDIF
    ENDIF
ENDPROCEDURE
```

When the subroutine is called for the first time, the arguments specified are 0 for `first` and 14 for `last`, i.e.

`search("James", scores, 0, 14)`

How many times in total will the subroutine be called (including the first call) so that the score for the student with the surname James can be retrieved?

All teaching materials on this site are available under a CC BY-NC-SA 4.0 license, except where otherwise stated.