Grade 11 Computing

Unit 2 · Recursion

ANSWER KEY · FOR TEACHERS

Full marks 33

Give the marks for the points in the marking scheme; an answer in other words that says the same is right.

1 Key words

1. recursive function
2. base case
3. recursive case
4. call stack
5. RecursionError
6. fractal
7. flood fill
8. merge sort

Each carries 0.5 mark. **K · 4**

2 Label the diagram

1. Start
2. Top 3 disks to B
3. Biggest disk to C
4. The 3 disks onto it

Each carries 0.5 mark. **U · 2**

3 Multiple choice

i. **c.** 10 **A · 1**

f(4) = 4 + f(3) = 4 + 3 + f(2) = 4 + 3 + 2 + 1 + f(0) = 4 + 3 + 2 + 1 + 0 = 10. It adds up the numbers from n down to 0. (24 would be 4! = 4 × 3 × 2 × 1.)

ii. **c.** 7 **A · 1**

factorial(6), (5), (4), (3), (2), (1) and (0): 7 calls, all open when factorial(0) starts, each waiting for the one above it. 720 is the answer, 6!, not a number of calls.

iii. **c.** It stops with a TypeError, because a call gives back None and None can't be multiplied. **A · 1**

factorial(1) works out 1 × 1 but doesn't return it, so it gives back None. factorial(2) then tries 2 × None: `TypeError: unsupported operand type(s) for *: 'int' and 'NoneType'`.

iv. **d.** 255 **A · 1**

$2^8 - 1 = 256 - 1 = 255$. Each extra disk doubles the moves and adds one: 7 for 3 disks, 15, 31, 63, 127, 255.

v. **c.** adding up the 100,000 numbers in a list **U · 1**

Adding up a list is a straight line, and with recursion a list of 100,000 numbers would need 100,000 open calls, far past the recursion limit. The other three branch or hold things inside things, where recursion is the clearer program.

4 What does it print?

a. **MARKING SCHEME** **A · 3**

- ✓ `down 3`, `down 2`, `down 1`: each call prints before it makes the next call.
- ✓ Nothing printed for `show(0)`, the base case.
- ✓ `up 1`, `up 2`, `up 3`: each call prints its second line only after the call it made has finished, so the last call made finishes first.

b. **MARKING SCHEME** **A · 2**

- ✓ `arahkoP`: each call puts its first letter after the rest of the string, reversed.
- ✓ 13: the digits added, 3 + 8 + 0 + 2.

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

5 Trace the calls

ANSWER A · 4

Call	n	half	Gives back
power(3, 5)	5	9	243
power(3, 2)	2	3	9
power(3, 1)	1	1	3
power(3, 0)	0	(none: the base case)	1

The program prints 243. The calls are made from top to bottom, and finish from the bottom up: `half` in each row is what the row below gave back.

ii. **11 calls:** n goes 1000, 500, 250, 125, 62, 31, 15, 7, 3, 1, 0. Multiplying by 3 one power at a time would take 1,000 multiplications.

MARKING SCHEME

- ✓ The calls in order, n = 5, 2, 1, 0 (each n halved, rounded down), with the base case last.
- ✓ What each call gives back: 1, then 3 (odd: 1 × 1 × 3), 9, 243.
- ✓ `half` in each row equal to what the next call gave back: 9, 3, 1.
- ✓ 11 calls for `power(3, 1000)`.

6 Count the calls

ANSWER A · 2

n	ways(n)	Calls of ways
0	1	1
1	1	1
2	2	3
3	3	5
4	5	9
5	8	15
6	13	25

Each `ways(n)` is the sum of the two above it. The calls are one for `ways(n)` itself plus all the calls of `ways(n - 1)` and of `ways(n - 2)`: for 6, 1 + 15 + 9 = 25.

MARKING SCHEME

- ✓ `ways(n)` 2, 3, 5, 8, 13.
- ✓ Calls 3, 5, 9, 15, 25.

7 Find and fix the bug

MARKING SCHEME A · 3

- ✓ A RecursionError.
- ✓ The square isn't colored yet when its neighbors are filled, so two neighbors keep calling each other and never reach a base case.
- ✓ `grid[r][c] = new` moved before the four calls.

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

8 Write a program

ANSWER **A · 4**

```
def total(items):
    t = 0
    for x in items:
        if isinstance(x, list):
            t = t + total(x)
        else:
            t = t + x
    return t

print(total([5, [10, [20, 3]], 2]), total([]), total([[[]], 7]))
```

```
OUTPUT
40 0 7
```

The base case is a list with no lists inside: the loop adds its numbers and makes no call. Also right: a version that adds the first item to `total(items[1:])`, with the empty list as the base case.

MARKING SCHEME

- ✓ A loop over the items (or the first item and the rest).
- ✓ A list found with `isinstance` (or `type(x) == list`) and its total found by calling `total` on it.
- ✓ A number added straight to the sum.
- ✓ The sum given back with `return`, 0 for an empty list.

9 Short answer

MARKING SCHEME **U · 2**

- ✓ The n − 1 smaller disks are moved out of the way and back on top (twice moves(n − 1)), and the biggest disk moves once (+ 1).
- ✓ moves(10) = 1,023 (= $2^{10} - 1$).

10 Think

ANSWER **HA · 2**

Each call asks two smaller questions, and nothing remembers an answer, so the same question is worked out again and again: ways(38) is worked out in full by ways(40) and again by ways(39), and the repeats multiply all the way down. The calls grow about as fast as the answers themselves. Remembering each answer the first time it is worked out (a dictionary of answers checked before working one out), or a loop that works up from ways(0) and ways(1) keeping the last two, works out each of the 41 values once, about 40 additions.

MARKING SCHEME

- ✓ The same smaller questions are worked out again and again, because nothing remembers an answer.
- ✓ Remember each answer (a dictionary), or use a loop from the bottom up, so each of the 41 values is worked out once.

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

11 Activity: Play the Tower of Hanoi

(not marked)

WHAT STUDENTS SHOULD FIND

The fewest possible are 7, 15 and 31: each one more than double the one before ($2^n - 1$ for n disks). The smallest disk moves every other turn, and always round the pegs the same way. Students who stop to ask "where must the smaller disks be when the biggest disk moves?" reach the fewest moves.

Marks by level	Knowledge	Understanding	Application	Higher ability	Total
Marks	4	5	22	2	**33**

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking