

THEME B: COMPUTATIONAL THINKING · B2.4

Programming algorithms

Original practice questions · 35 marks · about 60 minutes · covers B2.4.1–B2.4.5

Instructions. Answer all questions. The maximum mark for this paper is [35 marks]. The marks available are shown in brackets. Parts badged **HL** assess higher level only content; standard level students may leave these out. Questions that ask you to construct code may be answered in **either Java or Python**. State which language you are using. Marks are for the logic, not for perfect syntax.

1 Total: 5 marks

Big O notation is used to describe how an algorithm scales as the amount of data grows.

- (a) State the Big O time complexity of a linear search and of a binary search. [2]

- (b) Explain what it means for an algorithm's scalability if its time complexity is $O(n^2)$. [3]

2 Total: 7 marks

A program searches the sorted array below for the value 40.

The sorted array **values**

Index	0	1	2	3	4	5	6	7
Value	4	9	15	22	31	40	55	68

- (a) Copy and complete the trace table for a binary search for the value 40. [4]

low	high	mid	values[mid]

- (b) State how many comparisons a linear search would need to find 40 in this array. [1]

- (c) Explain why a binary search cannot be used unless the data is sorted. [2]

A bubble sort is applied to the array 5, 2, 9, 1. One pass means working through the array once, comparing each neighbouring pair and swapping them when they are out of order.

- (a) Copy and complete the table to show the array after each pass.

[4]

After pass	Array
start	5, 2, 9, 1

- (b) State the worst-case time complexity of a bubble sort.

[1]

- (c) Compare a bubble sort with a selection sort in terms of the number of swaps each performs.

[3]

A selection sort works by repeatedly finding the smallest remaining value and moving it into place.

Construct a program that sorts an array of integers into ascending order using a selection sort.

[6]

5

Total: 5 marks

Some problems are solved more naturally using recursion.

- (a) **HL** Explain what is meant by recursion, including why a base case is necessary. [3]

- (b) **HL** Outline **one** limitation of using recursion rather than a loop. [2]

6

Total: 4 marks

The recursive function below is written in Java and in Python.

JAVA

```
public static int total(int n)
{
    if (n == 0)
        return 0;
    else
        return n + total(n - 1);
}
```

PYTHON

```
def total(n):
    if n == 0:
        return 0
    else:
        return n + total(n - 1)
```

- (a) **HL** State the value returned by `total(4)`. [2]

- (b) **HL** State what would happen if the `if` branch were removed from the function. [2]

END OF QUESTIONS