2.1 Algorithms: Test Pack

OCR J277 · Calculators not allowed · about 20 minutes

Total Marks

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 Computational thinking involves breaking down complex problems using specific techniques before an algorithm is designed.

(a) Define what is meant by 'decomposition' when designing a solution to a problem. **(2)**

(b) Define what is meant by 'abstraction' when designing an algorithm. **(2)**

(Total for Question 1 is 4 marks)

2 Flowcharts use standard symbol shapes to represent different steps in an algorithm.

(a) State the flowchart symbol shape used to represent a decision. **(1)**

(b) State the flowchart symbol shape used to represent an input or output step. **(1)**

(c) State the flowchart symbol shape used to represent the start or end (terminator) of an algorithm. **(1)**

(Total for Question 2 is 3 marks)

© Revision Library | revisionlibrary.co.uk

3 A programmer writes the following pseudocode.

```
count <- 0
num <- 18
WHILE num > 1
    num <- num / 2
    count <- count + 1
ENDWHILE
OUTPUT count
```

(a) Complete a trace table showing the values of num and count each time the loop body executes. **(5)**

(b) State the number of times the loop body executes. **(1)**

(Total for Question 3 is 6 marks)

---

4 An array called numbers contains 6 elements, at indices 0 to 5.

(a) Write an algorithm, using pseudocode, for a bubble sort that sorts the array numbers into ascending order. Your algorithm should use nested loops. **(6)**

(Total for Question 4 is 6 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 2.1 Algorithms: Test Pack

Question 1

(a) identifies decomposition as breaking a large or complex problem down (oe) **B1**

(a) into smaller, more manageable sub-problems that are easier to solve individually (oe) **B1**

(a) Answer: Decomposition is the process of breaking a large, complex problem down into a series of smaller, more manageable sub-problems.

(b) identifies abstraction as removing or ignoring unnecessary detail (oe) **B1**

(b) so that only the information relevant to solving the problem is kept/used (oe) **B1**

(b) Answer: Abstraction is the process of removing unnecessary detail from a problem so that only the information relevant to solving it remains.

Question 2

(a) diamond (accept rhombus) **B1**

(a) Answer: Diamond (rhombus).

(b) parallelogram **B1**

(b) Answer: Parallelogram.

(c) oval/rounded rectangle (terminator) **B1**

(c) Answer: Oval (rounded rectangle).

Question 3

(a) num = 9, count = 1 **B1**

(a) num = 4.5, count = 2 **B1**

(a) num = 2.25, count = 3 **B1**

(a) num = 1.125, count = 4 **B1**

(a) num = 0.5625, count = 5 (all five rows correct, cao) **B1**

(a) Answer: num: 9, 4.5, 2.25, 1.125, 0.5625 / count: 1,2,3,4,5

(b) 5 (cao) **B1**

(b) Answer: 5

Question 4

(a) outer loop that repeats a suitable number of times for the array size, e.g. FOR i <- 0 TO 4 **M1**

(a) inner loop that iterates through adjacent pairs of elements, e.g. FOR j <- 0 TO 4 - i **M1**

(a) correct comparison of adjacent elements, e.g. IF numbers[j] > numbers[j + 1] THEN **M1**

(a) uses a temporary variable to hold one value during the swap, e.g. temp <- numbers[j] **M1**

(a) both assignments of the swap are correct, e.g. numbers[j] <- numbers[j+1] and numbers[j+1] <- temp **M1**

(a) overall algorithm correctly sorts the array into ascending order with no logic errors (any correct logic; ignore minor syntax errors) **A1**

(a) Answer:

```
FOR i <- 0 TO 4
    FOR j <- 0 TO 4 - i
        IF numbers[j] > numbers[j + 1] THEN
            temp <- numbers[j]
            numbers[j] <- numbers[j + 1]
            numbers[j + 1] <- temp
        ENDIF
    NEXT j
NEXT i
```


© Revision Library | revisionlibrary.co.uk