51 /51

IGCSE Computer Science

Name: [redacted]

Chapter 7 & 8 Test: Algorithms and programming

Q1

Write an algorithm to input 1000 numbers. Count how many numbers are positive and how many numbers are zero. Then output the results. Use **either** pseudocode **or** a flowchart.

```
~~WHILE count <1000~~ ~~DECLARE zeros : INTEGER~~
zeros ← 0
pos ← 0
count ← 1
WHILE count <= 1000 DO
    ~~INPUT number~~
    DECLARE number : INTEGER
    INPUT number
    ~~CASE OF number~~ ~~IF number = 0 THEN~~
    IF number = 0 THEN
        ~~THEN~~ zeros ← zeros + 1
    ELSE IF number > 0 THEN
        pos ← pos + 1
    ENDIF
    count ← count + 1
ENDWHILE
~~Output~~ OUTPUT zeros, pos
```

6/6 [6]

Give one change you could make to your algorithm to ensure initial testing is more manageable.

Change the number of numbers entered to a lower value

e.g. 50 numbers [1]

Q2

The global trade item number (GTIN-8) barcode has seven digits and a check digit.
This pseudocode algorithm inputs seven digits and calculates the eighth digit, then outputs the GTIN-8.

`DIV(X,Y)`, finds the number of divides in division for example `DIV(23,10)` is 2.
`MOD(X,Y)`, finds the remainder in division for example `MOD(23,10)` is 3.

```
FOR Count ← 1 TO 7
   INPUT Number
   Digit(Count) ← Number
NEXT
Sum ← (Digit(1)+Digit(3)+Digit(5)+Digit(7))*3+Digit(2)+Digit(4)+Digit(6)
IF MOD(Sum,10) <> 0
   THEN Digit(8) ← DIV(Sum,10)*10 + 10 - Sum
   ELSE Digit(8) ← 0
ENDIF
OUTPUT "GTIN-8"
FOR Count ← 1 TO 8
   OUTPUT Digit(Count)
NEXT
```

4 0 1 0 - 4 4 = 6

5 + 0 + 2 + 4

33 + 7 + 1 + 3

= 44

(a) Complete the trace table for the input data: 5, 7, 0, 1, 2, 3, 4

Digit(1)	Digit(2)	Digit(3)	Digit(4)	Digit(5)	Digit(6)	Digit(7)	Digit(8)	Sum	OUTPUT
5	7	0	1	2	3	4		44	
							6		GTIN-8 5 7 0 1 2 3 4 6

4 + 1 + 2 + 1 = 8

24 + 3 + 0 + 3 = 30

Complete the trace table for the input data: 4, 3, 1, 0, 2, 3, 1

Digit(1)	Digit(2)	Digit(3)	Digit(4)	Digit(5)	Digit(6)	Digit(7)	Digit(8)	Sum	OUTPUT
4	3	1	0	2	3	1		30	
							0		GTIN-8 4 3 1 0 2 3 1 0

[5]

Q3

Tick (✓) **one** box in each row to identify if the statement about structure diagrams is true or false.

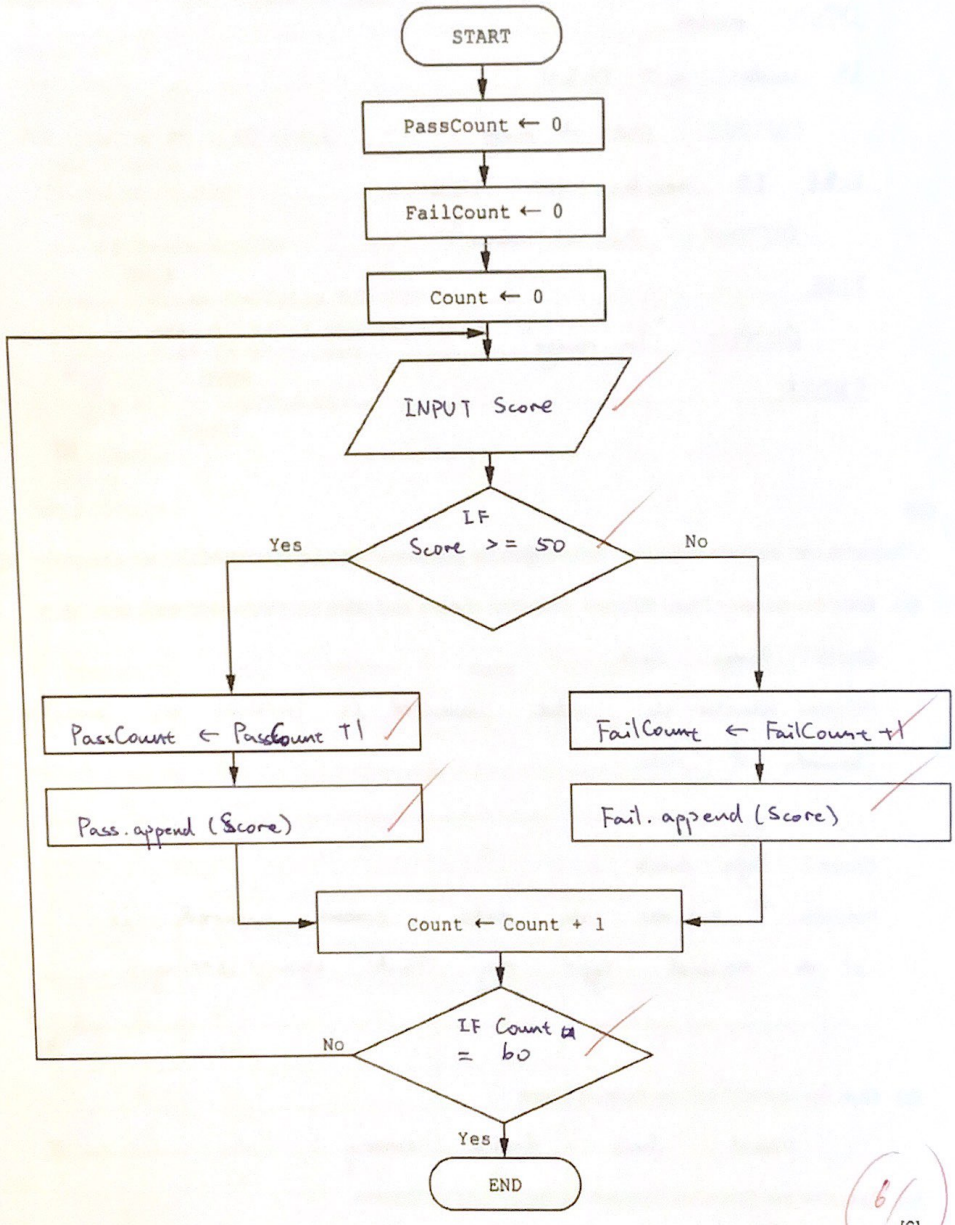
Statement	True (✓)	False (✓)
A structure diagram is a piece of code that is available throughout the structure of a program.		✓
A structure diagram shows the hierarchy of a system.	✓	
A structure diagram is another name for an array.		✓
A structure diagram shows the relationship between different components of a system.	✓	

[2]

Q4

4 The flowchart shows an algorithm that should allow 60 test results to be entered into the variable `Score`. Each test result is checked to see if it is 50 or more. If it is, the test result is assigned to the `Pass` array. Otherwise, it is assigned to the `Fail` array.

(a) Complete this flowchart:



[6]

Q5

Write a pseudocode routine that will check that each test result entered into the algorithm is between 0 and 100 inclusive.

```
DECLARE results : INTEGER FLOAT
INPUT results
IF results < 0 THEN
    OUTPUT "Out of range"
ELSE IF results > 100 THEN
    OUTPUT "Out of range"
ELSE
    OUTPUT "In range"
ENDIF
```

4 [4]

Q6

Programs can perform validation and verification checks when data is entered.

(a) Give the names of **two** different validation checks and state the purpose of each one.

Check 1 Range check

Purpose Whether the value entered is within the possible bounds of data

Check 2 Type check

Purpose Whether the data entered is of the required type e.g. float, char, string.

4 [4]

(b) Give the name of **one** verification check.

Visual check / double entry 1 [1]

(c) Describe the difference between validation and verification.

Verification is where the user checks whether the data entered is what they wanted to be entered, validation is where a program checks whether the data entered meets the requirements of the data [2]

2/2

Q7

The pseudocode represents an algorithm.

The pre-defined function `DIV` gives the value of the result of integer division.
For example, `Y = 9 DIV 4` gives the value `Y = 2`

The pre-defined function `MOD` gives the value of the remainder of integer division.
For example, `R = 9 MOD 4` gives the value `R = 1`

```
First ← 0
Last ← 0
INPUT Limit
FOR Counter ← 1 TO Limit
  INPUT Value
  IF Value >= 100
    THEN
      IF Value < 1000
        THEN
          First ← Value DIV 100
          Last ← Value MOD 10
            IF First = Last
              THEN
                OUTPUT Value
            ENDIF
      ENDIF
  ENDIF
NEXT Counter
```

(a) Complete the trace table for the algorithm using this input data:

8, 66, 606, 6226, 8448, 642, 747, 77, 121

Counter	Value	First	Last	Limit	OUTPUT
0		0	0	8	
1	66				
~~2~~	606				
2		6	6		606
3	6226				
4	~~6226~~8448				
~~5~~	642	~~6~~	~~2~~		
5	~~8~~	6	2		
	747				
6		7	7		747
7	77				
	121	~~7~~	~~7~~		

8 ✓ 1 1 121 ✓ [5]

Q8

A program stores the ID number of items for sale in a shop and the price per item. This is stored in a 2-dimensional array of 100 elements with the identifier `items`. An example is given:

Index	0	1	2	3
ID (0)	10.0	11.1	20.3	6.4
Price (1)	5.50	1.99	35.00	26.52

The program needs to take an item ID and quantity from the user. If the ID is valid it calculates the cost of that item and adds it to a total cost for an order. If the ID is invalid it outputs an appropriate message.

The program should total the number of individual items bought, for example, if they buy 20 of item ID 10.0, and 5 of item ID 11.1 then it will total 25 items.

The program needs to continually take IDs from the user until they enter a command to stop. The program then outputs the total cost of the order with an appropriate message.

Write the program to perform the tasks given. You do not need to initialise the values in the array `items`.

```
total = 0                                   # defining variables
bought = []
bought_ID = []
buy = True
found = False
while buy != False:
    ID = float(input('Please input the ID of the item you want to buy,
                     and if you don't want to buy anything, enter
                     the number -1 \n>'))   # input of IDs
    if ID != -1.0:                          # Whether the user wants to buy or quit
        for i in range(len(items)):         # Finding the ID in the array
            if ID == items[i][0]:
                bought_ID.append(ID)        # Creating list of IDs
                bought.append(items[i][1])  # Creating list of costs
                found = True
        if found == False:
            print('Not found')              # Error message when ID is not found
    else:
        buy = False                         # don't want to buy more
```

```
total = sum(bought)                                    # total cost
comb = [bought.ID.count(j) for j in bought.ID]         # times each ID is bought
print('The total cost is ${0}, and the items bought is {1}'.format(total, comb))
```

Excellent.

15/

15/ [15]