The City School

Unified Mid-Year Examinations

2018 - 2019

Class 11

I AM · TO LEARN
Since 1978

SCHOOL NAME			
INDEX NUMBER	□□□□	**DATE**	

COMPUTER SCIENCE **2210/22**

Paper 2 Problem-solving and Programming **1 hour 45 minutes**

Candidates answer on the Question Paper.
No Additional Materials are required.

READ THESE INSTRUCTIONS FIRST

Write your School name, Index number and Date in the spaces provided.
Write in dark blue or black pen.
You may use a pencil for any diagrams, graphs or rough working.
Do not use paper clips, glue or correction fluid.
Calculators must not be used in this paper.

Answer **all** questions.

DO NOT ATTEMPT TASKS 1, 2 AND 3 in the pre-release material; these are for information only.

You are advised to spend no more than **40 minutes** on **Section A** (Question 1).

No marks will be awarded for using brand names of software packages or hardware.

At the end of examination, fasten all your work securely together.
The number of marks is given in brackets [ ] at the end of each question or part question.
The maximum number of marks is 50.

Invigilated By: ___________ Checked By: ___________ Marks Tallied By: ___________

This document consists of **10** printed pages and **2** blank pages.

Section A

You are advised to spend no longer than 40 minutes answering this section.

Here is a copy of the pre-release material.

DO NOT attempt Tasks 1, 2 and 3 now.

Use the pre-release material and your experience from attempting the tasks before the examination to answer Section A Question 1.

Pre-release material

A teacher needs a program to record marks for a class of 30 students who have sat three computer science tests.

Write and test a program for the teacher.

- Your program must include appropriate prompts for the entry of data.
- Error messages and other output need to be set out clearly and understandably.
- All variables, constants and other identifiers must have meaningful names.

You will need to complete these three tasks. Each task must be fully tested.

TASK 1 – Set up arrays

Set-up one dimensional arrays to store:

- Student names
- Student marks for Test 1, Test 2 and Test 3
    - Test 1 is out of 20 marks
    - Test 2 is out of 25 marks
    - Test 3 is out of 35 marks
- Total score for each student

Input and store the names for 30 students. You may assume that the students' names are unique.

Input and store the students' marks for Test 1, Test 2 and Test 3. All the marks must be validated on entry and any invalid marks rejected.

TASK 2 – Calculate

Calculate the total score for each student and store in the array.

Calculate the average total score for the whole class.

Output each student's name followed by their total score.

Output the average total score for the class.

TASK 3 – Select

Select the student with the highest total score and output their name and total score.

1 **(a)** All variables, constants and other identifiers should have meaningful names.

(i) Declare the data structure to store the students' names.

....

.... [1]

(ii) Declare the data structure to store each student's marks and total score.

....

.... [2]

(b) (i) Show the design of your algorithm to complete Task 1 and Task 2 using
pseudocode, programming statements or a flowchart. Do not include any
of the validation checks or input prompts in your algorithm.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [8]

(ii) Comment on the efficiency of your design.

....

....

.... [1]

(c) Show two different sets of student data that you could use to check the validation used in Task 1. Explain why you chose each data set.

Set 1 ....

Reason for choice ....

....

Set 2 ....

Reason for choice ....

.... [2]

(d) (i) Explain how you select the student with the highest score (Task 3). You may include pseudocode or programming statements to help illustrate your explanation.

....

....

....

....

....

....

....

....

....

.... [5]

(ii) How does your program work when there is more than one student having the highest score? Explain using your method given in part (d)(i).

....

....

.... [1]

Section B

2 **(a)** Write an algorithm to input 500 numbers. Count how many numbers are **negative**, how many numbers are **positive** and many numbers are **zero**. Then output the results. Use **either** pseudocode **or** a flowchart.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [6]

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

....

.... [1]

3 Explain the difference between the programming concepts of **counting** and **totaling**. You may include an example of a programming statement for each concept in your explanation

....

....

....

....

....

....

....

....

....

....

....

....

.... [2]

4 The following section of the pseudocode inputs 10 temperature readings; calculate and print lowest, highest and average temperature recorded.

```
lowest ←1000
highest← 0
sum←0
averagetemperature←0

For a ←1 to 10

INPUT temperature

IF temperature < lowest Then lowest ←temperature
IF temperature > highest Then highest ←temperature

sum←sum+temperature

Next a

averagetemperature = sum/10
PRINT lowest, highest, averagetemperature
```

(a) Complete the trace table for the input data: 20, 10, 15, 11, 5, 7, 21, 28, 23, 17

temperature	lowest	highest	sum	averagetemperature	OUTPUT

[5]

(b) Draw the **flowchart** of the **pseudocode** of part (a)

[4]

5 The following section of the pseudocode

- input a positive integer
- use this value to set up how many other numbers are to be input
- input these numbers
- calculate and output the total and the average of these numbers

Locate the **3 errors** and suggest a corrected piece of code

```
1  INPUT NumberCount
2      Total←0
3          FOR Count←1 TO NumberCount
4      INPUT Number
5     Total←Total + 1
6   Average←Total/NumberCount
7  NEXT Count
8  PRINT NumberCount
```

Error 1

....

.... [1]

Error 2

....

.... [1]

Error 3

....

.... [1]

6 A database table, **INVENTORY** is used to keep records of CPUs in IT Lab

Service Tag No	Name of Device	Model	Status	Under Warranty	Type of socket	Date of Purchase
A01	CPU	SM1	OK	N	LGA	21-JUN-15
A02	CPU	SX	OK	Y	PGA	01-AUG-15
A03	CPU	Z1	DEAD	Y	BGA	12-SEP-17
A04	CPU	SX	DEAD	Y	LGA	23-FEB-18
A05	CPU	AE1	OK	Y	PGA	29-DEC-17
A06	CPU	AFF	OK	Y	LGA	22-MAR-15

(a) State the number of fields and records in the table.

Fields ....

Records .... [2]

(b) State the suitable field for primary key in the table.

.... [1]

(c) Give **two** validation checks that could be performed on the **Service Tag No** field.

Validation check 1 ....

....

Validation check 2 ....

.... [2]

(d) Using the query-by-example grid, write a query to identify CPUs that are **under warranty** and **status** is **dead**. Only display the **Service Tag No** and **Model**

Field :					
Table:					
Sort:					
Show:	□	□	□	□	□
Criteria:					
Or:					

[4]