GENERAL CERTIFICATE OF EDUCATION BOARD

General Certificate of Education Examination

0795 Computer Science 3

JUNE 2022 **ADVANCED LEVEL**

Subject Title	**Computer Science**
Paper No.	**Paper 3 – Practical**
Subject Code No.	**0795**

Two Hours

Carry out ALL the tasks given. For your guidance, the approximate mark for each part of a task is indicated in brackets.

Great importance is attached to the accuracy, layout and labelling of the drawings and computer generated outputs.

You are reminded of the necessity for good English and orderly presentation of your answers.

Write algorithms in the answer booklet provided. Also record in your answer booklet any information requested or that you believe would make it easier to understand how you carried out tasks or answered questions.

You are expected to print out a single copy of relevant fragments of your program at different times. ***Please notify the instructor of any required printout that was not done!***

When an imperative programming language is required to write program code, either **Standard [ISO]** Pascal or the **[ANSI] C or C11** programming languages may be used.

If need be, supervisors will assist you in recording details of intermediate work carried out on the computer.

Do not write on the first page of your answer booklet. It is reserved for administrative purpose.

Where information is provided as soft copy, notify the instructors if it is not found in your machine or has not been made available to you.

Turn Over

Task 1: Database for a Furniture Company **(24 marks)**

A company sells furniture to customers of its store. The store does not keep furniture in stock. Instead, a customer places an order at the store and the company then orders the furniture required from its suppliers. When the ordered furniture arrives at the store a member of staff telephones or emails the customer to inform him/her that it is ready for collection. Customers often order more than one type of furniture on the same order, for example a sofa and two chairs.

Details of the furniture, customer and orders are to be stored in a relational database using the following relations:

Furniture(<u>FurnitureID</u>, FurnitureName, Category, Price, SupplierName)
CustomerOrder(<u>OrderID</u>, CustomerID, Date)
CustomerOrderLine(<u>OrderID</u>, <u>FurnitureID</u>, Quantity)
Customer(<u>CustomerID</u>, CustomerName, EmailAddress, TelephoneNumber)

The attributes for these relations and others used eventually, are detailed as follows:

Description	Data type	Field size	Remark
FurnitureID	Number	6	Must be present
FurnitureName	Varchar	15	
Category	Varchar	15	Must be present
Price	Number		Must be present
SupplierName	Varchar	10	
OrderIdD	Number		Must be present
Date	Date		
Quantity	Number		
CustomerID	Varchar	6	Must be present
CustomerName	Varchar	15	
Email	Varchar	20	
Telephone number	Number		
Validation	Text		

You are to develop a database for the furniture company via the subtasks given below.

(i) In your answer booklet, give an Entity-Relationship (E-R) diagram that shows the degree of any **three** relationships that exist between the above entities. **(3 marks)**

(ii) Using SQL statements, do the following using your favourite database management system (DBMS)
1. Create a database and call it **SELLS**. **(1 mark)**
2. In **SELLS**, create all the relations given above. **(4 marks)**

(iii) Populate the relations in your database with 5-7 rows of furniture data and 3-4 rows of customer data. The data entered for CustomerOrder and CustomerOrderLine should be consistent with those for relations Furniture and Customer. That is, their data should derive from the furniture purchase transactions that the customers make. **(8 marks)**

(iv) [**Note**: *For this subtask, first write in your answer booklet the SQL queries (or statements) used to implement its activities in your DBMS.*]
Suppose a fault has been identified with the product whose FurnitureID is 10765. In order to contact all those who purchased this item of furniture, the manager of the company needs a list of their <u>names and telephone numbers only</u>. Also, the list should be in alphabetical order of customer names.
(a) Give an SQL query that produces this list. **(2 marks)**
(b) Write an SQL query to delete from the database the faulty product with furnitureID number 10765. **(2 marks)**

(c) Alter the table Customer Order to include a field called **Validation**. The field holds a short explanatory text for faults found in an order, or the word "NONE" if there is no fault. **(2 marks)**

(d) In your answer booklet, mindful of how you produced the list of names and telephone numbers in iv(a) above, state in words how one would remove faulty products from your database. **(2 marks)**

Task 2: Programming **(26 marks)**

You are to design a program to help pupils in primary school to list all the prime numbers found within a given range of positive integers. A prime number is divisible only by one and itself. However, 1 is not considered prime. Use the following algorithm to get the desired list of prime numbers.

1. Get the maximum value of a range from the user. Call it **N**.
2. Fill the array with numbers from 1 to **N**. Take your array size to be 100.
3. Starting from the beginning of the array, eliminate the number 1 from the array by setting it to zero. Then, locate the first prime number.
4. Set all its multiples to zero—except, of course, the number itself—and then move to the next number.
5. Repeat step (4) for each new number there, until you get to the square root of **N**. Call this square root **rootN**. Obtain it via the square root function of your programming language (PL), and round it up (or down) to an integer value. [**NOTE**: *Rounding may be either through a* ***round*** *function (or whatever your PL environment might offer) or truncation to an integer value such as through a type cast, integer arithmetic or assignment to integer variables*.]
6. Display all the numbers in the array that are different from zero. They should all be prime numbers..

Illustration:

Here, we illustrate how the algorithm works.

1. You obtain (declare) an array **Tab[100]** of integer.
2. Suppose the maximum value of the range given is **N**=10.
3. Fill array **Tab** with **N** numbers.

1	2	3	4	5	6	7	8	9	10

4. Set entry 1 (the first) to zero.

0	2	3	4	5	6	7	8	9	10

Now, we identify all the prime numbers from 2 to **rootN** and set to zeros their multiples found in the array as follows:

5. Round up the square root of **N** (i.e., of 10) to get 4.
6. Identify 2 as a prime number.
7. Excluding 2 itself, set to zero all multiples of 2, right up to N. So, we reset the numbers 4, 6, 8 and 10 to obtain:

0	2	3	0	5	0	7	0	9	0

8. Identify the next prime number (i.e. 3).
9. Again, excluding 3, we set all the multiples of 3 (i.e., 3 and 9) to zero to obtain:

0	2	3	0	5	0	7	0	0	0

10. The next available [non-zero] number, 5, is greater than **rootN** (i.e., 4) so we stop. We now display (or print) all the non-zero numbers of the array as the prime numbers found between 1 and 10, viz: 2, 3, 5 and 7.

Application:
Here, we develop the required program by following the steps below.

(i) In your answer booklet, state the programming language you will use. Then, give a more detailed version of the algorithm above in pseudo code. In it, let **N** = 20. **(3 marks)**

(ii) Write a subprogram called **initialise(tab, n)** that fills the array **tab** with numbers from 1 to **n**. **(4 marks)**

(iii) Write a subprogram called **verifyprime(x)** that takes a number **x** and verifies whether it is a prime number or not. For this, **x** is not divisible by any non-zero number less than it in the array **tab.** **(5 marks)**

(iv) Write a subprogram called **setzero(tab, n, y)** that takes a number **y** and sets all its multiples in array **tab** to zero, except the number itself, as explained earlier. **(4 marks)**

(v) Write a subprogram called **doDisplay(tab, n)** that displays all the non-zero elements of array **tab**, up to those in array position **n**. **(3 marks)**

(vi) Write a main program that makes use of the above subprograms to identify prime numbers in a positive integer range. Make sure your program works correctly. **(4 marks)**

(vii) Print your entire program including its subprograms. **(1 mark)**

(viii) Test your program with **N**= 100. Save its output and then print it. **(2 marks)**