REGISTRATION CENTRE NUMBER	CENTRE NAME
CANDIDATE'S FULL NAMES	

CANDIDATE IDENTIFICATION NUMBER	SUBJECT CODE 0795	PAPER NUMBER 3

FOR OFFICIAL USE ONLY (Candidate Random CODE): ►	

HOLY INFANT HIGH SCHOOL – MELEN

ADVANCED LEVEL MOCK EXAMINATION

SUBJECT TITILE COMPUTER SCIENCE	SUBJECT CODE 0795	PAPER NUMBER 3
	EXAMINATION DATE:	**MARCH 2024**

FOR EXAMINERS' USE ONLY	
Marked By: …….. Signature: …….. Date: …….....	**SCORE**
Checked By: …….. Signature: …….. Date: …….....	

TWO HOURS

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

Great importance is attached to the accuracy, layout and labeling of diagrams and computer-generated outputs.

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

You are expected to print a single copy of relevant fragments of your work at different times. Please notify the supervisor 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 Standard (ANSI) C programming language may be used.
If need be, supervisors will assist you in recording details of intermediate work carried out on the computer.

PROGRAMMING: ELECTRIC MOUNTAIN RAILWAY SYSTEM

An electric mountain railway makes four return trips every day. In each trip the train goes up the mountain and back down. The train leaves from the foot of the mountain at 09:00, 11:00, 13:00 and 15:00. The train returns from the top of the mountain at 10:00, 12:00, 14:00 and 16:00. Each train has **SIX** coaches with **EIGHTY** seats available in each coach. Passengers can only purchase a return ticket; all tickets must be purchased on the day of travel. The cost is $25 for the journey up and $25 for the journey down. Groups of between ten and eighty passengers inclusive get a free ticket for every tenth passenger, provided they all travel together (every tenth passenger travels free). Passengers must book their return train journey, as well as the departure train journey, when they purchase their ticket. Passengers can return on the next train down the mountain or a later train. The last train from the top of the mountain has two extra coaches on it.
The train times are displayed on a large screen, together with the number of tickets still available for each train. Every time a ticket is booked the display is updated. When a train is full, the word 'Closed' is displayed instead of the number of tickets available.
The train times are displayed on a large screen, together with the number of tickets still available for each train. Every time a ticket is booked the display is updated. When a train is full, the word **'Closed'** is displayed and **'Opened'** displayed otherwise.

Task 1 – Start of the day.

At the beginning of the day, you are to set up the screen display for the start of the day for all the Eight journeys. You are to use suitable data structure(s) to total passengers for each train journey and total the money taken for each train journey. Each train journey must be totaled separately. There are four journeys up and four journeys down every day.
All variables, constants and other identifiers must have meaningful names. They must be declared globally. A sample code has been given to you by the supervisor which has contains all the variables which you may need for the smooth running of the programs. You may add or modify it to fit your needs.

(a) Write a program that displays the available tickets for the day. The pseudo code Procedure ***Display ( )*** for the screen to display the available tickets is found in the appendix (Page 5).
Save the source code as ***Task 1*** **(4 marks)**

(b) Run your program and do a screenshot. Save the screenshot as ***Task 1a*** **(1 mark)**

Task 2 – Purchasing tickets.

Tickets can be purchased for a single passenger or a group. When making a purchase, check that the number of tickets for the required train journeys up and down the mountain is available. If the tickets are available, calculate the total price including any group discount. Update the screen display and the data for the totals.

(a) State and explain **A** validation checks that could be used when inputting the number of tickets to buy for **Task 2**. For each validation check give one example of normal data and one example of erroneous data.

Validation check ……
Normal data .....
Erroneous data .....
(2 marks)

(b) Write a procedure ***BuyTickets ( )*** in a programming language that inputs the tickets required, calculates the total price for the ticket(s) purchased by a passenger including a group discount (if applicable) and updates the data structures used for the totals. Use the pseudocode found in the appendix to write the program. **(3 marks)**

(c) Apply the Validation checks mentioned in (a) above. Refer to the template file ***Sample.c*** **(4 marks)**

(d) Save your program as ***Task 2*** **(1 mark)**

(e) Run the program. Screenshot the output and Save it as ***Task 2a.*** **(2 marks)**

Task 3 – End of the day.

(a) Write a Procedure ***EndofDay ( )*** which will display the number of passengers that travelled on each train journey, the total money taken for each train journey, calculate and display the total number of passengers, the total amount of money taken for the day; find and display the train journey with the most passengers that day.
(6 marks)

(a) Save your source code as ***Task 3.c*** *and* ***Print it*** **(1 mark)**

(b) Run the Program, screenshot the output and Save as ***Task 3a***. **(2 marks)**

DATABASE: LIBRARY MANAGEMENT SYSTEM

An after-school club runs different courses and has specialist teachers teaching one course only. Whereas one student can choose to attend various courses.

Details of the Student, Course, Teacher and CourseEnrolled are stored in a relational database using the following four relations:

Student (StudentId, ForeName, SurName, DOB, Address, City)
Course (CourseId, Title, Duration, Fee, CourseType)
Teacher (TeacherId, ForeName, SurName, CourseId)
CourseEnrolled (StudentId, CourseId)

In this system, the following restrictions apply to some attributes:

- CourseId: a mixture of **exactly** 6 letters and numbers, e.g., Step01
  - Same rule applies to other IDs.
- Duration: a whole number representing the length of the course e.g., 33
- CourseType: can be either 'Weekly' or 'Monthly' and nothing else
- Fee: a monetary value, eg 100

(a) Write an SQL statement to create a database named ***CLUB*** using any suitable software package. **(1 mark)**

_____

(b) Create the table ***Student, Course, Teacher, and CourseEnrolled***.
Write the SQL statement to create the table ***Course*** in your answer booklet. **(2 marks)**

_____
_____
_____
_____
_____

(c) Create a relationship diagram between all the tables.
Screenshot the relationships in the database and save it as ***Relationships.*** **(2 marks)**

(d) Do two records for each of the relations. **(6 marks)**
Write the SQL statement to insert two records in the table ***Course*** in your answer booklet.

_____
_____
_____

(e) The school has introduced another course for GCE-Computer-FT which is fast track i.e., can be done in one year with a fee of 5000FCFA and has **CourseId** GCEC02.
Write the SQL statement to add this new course into Course table. Save the query as ***New***. **(3 marks)**

CourseID: GCEC02 Title: GCE-Computer-FT Duration: 12

Fee: 5000 Course type: Monthly

_____
_____
_____
_____

(f) After above change, now the course with name GCE-Computer-FT has been changed to GCE-Computer-ST. Write the SQL statement to update the data in the Course table to reflect this change. Save query as ***Update***. **(3 marks)**

(g) The school wants to have a list of students with their forename and surname who attend weekly courses taught by a teacher William Smith. The result should in ascending order of students' forenames. Write an SQL query to achieve this. **(4 marks)**

(h) The school has now decided to incorporate parents database into this structure. Most of the children have their siblings enrolled in the school as well. The database must keep a record of all the parents who have their children enrolled in the school.

a) Explain how you would change the design of the database so that the information about parents can be stored. **(3 marks)**

b) Write the relation(s) created in 17(a) above. **(3 marks)**

Print all your Queries, Tables and Relationship diagram.

Appendix

Procedures (pseudocode)

```
//Procedure Display ()
Procedure Display ()

  //display train at the top

  //for loop for display of Train for the Journey to the Top of the Mountain
  For i ← 0 to 3
    //check availability of the train
    If (TseatsUp[i] == 0) Then
        StatusUp[i] ← "Closed"

    EndIf

    //Display the elements of the Train at the Top of the Mountain
    Output i+1,TtimeUp[i],TseatsUp[i],OccUpseat[i],AmtUpTrain[i],StatusUp[i]
  EndFor

  Goto new line


  For i ← 0 to 3
    //check availability of the train
    If (TseatsDown[i] == 0) Then
        StatusDown[i] ← "Closed"
    EndIf

    //Display the elements of the Train at the Foot of the Mountain
    Output i+1,TtimeDown[i],TseatsDown [i],OccDownseat[i],AmtDownTrain[i],StatusDown[i]
  EndFor
EndProcedure
```

```
//Procedure BuyTicket ()
Procedure BuyTickets ()

    Output "Do you want to buy Tickets? (Yes/No)"
    Input temp

    If (temp == "Yes" OR temp == "yes" OR temp = "YES")
        choice ← true

    Else
        choice ← false
    EndIf
    While choice <> false
        //Buy Tickets for the Journey to the Top
        Output "Enter the train num to journey to the Top [1,2,3,4]:"
        Input UpTrip
        Decrement UpTrip
        //Put a code to verify if train is full below

        //Input number of passengers
        Output "Enter number of passengers for the trip:"
        Input NumPass
```

```
        //Put Code to Verify if the number of passengers entered is in the correct below

        OccUpseat [UpTrip] ← OccUpseat [UpTrip] + NumPass
        TotalUpSeats [UpTrip] ← TotalUpSeats [UpTrip] - NumPass
        Cost ← NumPass * CostTicket
        //Discount of every 10th passenger
        Discount ← cost-(CostTicket * (NumPass/10))
        //Total Amount paid for the Trip Up
        AmtUpTrain [UpTrip] ← AmtUpTrain [UpTrip] + Discount

        //Buy Tickets for the Journey to the Foot
        Output "Enter the train num to journey to the Foot [1,2,3,4]:"
        Input DownTrip
        Decrement DownTrip

        //Put a code to verify if train is full below

        //Input number of passengers
        Output "Enter number of passengers for the trip:"
        Input NumPass

        //Put Code to Verify if the number of passengers entered is in the correct below


        OccDownseat [DownTrip] ← OccDownseat [DownTrip] + NumPass
        TotalDownSeats [DownTrip] ← TotalDownSeats [DownTrip] - NumPass
        Cost ← NumPass * CostTicket
        //Discount of every 10th passenger
        Discount ← cost-(CostTicket * (NumPass/10))
        //Total Amount paid for the Trip Down
        AmtDownTrain [DownTrip] ← AmtDownTrain [DownTrip] + Discount

        //You have to buy tickets for up trip and down trip
        TwoWayTicket ← Discount*2

        Clear screen (use system("cls") in C)
        Call Display ()
        Output "Do you want to buy Tickets? (Yes/No)"
        Input temp

        If (temp == "Yes" OR temp == "yes" OR temp = "YES")
            choice ← true

        Else
            choice ← false
        EndIf
    EndWhile

EndProcedure
```