REPUBLIQUE DU CAMEROUN
Paix-Travail-Patrie

MINISTERE DES ENSEIGNEMENTS SECONDAIRES

CELLULE D'APPUI A L'ACTION PEDAGOGIQUE
ANTENNE RÉGIONALE DU NORD OUEST

BP 2183 MANKON BAMENDA
TEL 233 362 209
Email : trubamenda@yahoo.co.uk

TEACHERS' RESOURCE UNIT TRU-RIP-STA
MOCK EXAMS
CAMP-BAMENDA NORTH WEST REGION

A
ATVE

REPUBLIC OF CAMEROON
Peace-Work-Fatherland

MINISTRY OF SECONDARY EDUCATION

TEACHERS' RESOURCE UNIT
REGIONAL BRANCH FOR THE NORTH WEST

P.O. BOX: 2183 MANKON BAMENDA
TEL 233 362 209
Email : trubamenda@yahoo.co.uk

MARCH 2023

The Teachers' Resource Unit and the Regional Inspectorate of Pedagogy, in collaboration with Computer Science Teachers' Association(COSTA	SUBJECT CODE NUMBER **0795**	PAPER NUMBER **3**
GENERAL CERTIFICATE OF EDUCATION AND INTERMEDIATE TECHINICAL AND VOCATIONAL EDUCATION REGIONAL MOCK EXAMINATION	SUBJECT TITLE **COMPUTER SCIENCE**	
ADVANCED LEVEL		

Time Allowed: **TWO hours**

INSTRUCTIONS TO CANDIDATES

Mobile phones are **NOT ALLOWED** in the examination room.

- *Answer all Questions.*
- *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 accuracy, layout and labeling of drawings and computer-generated outputs.
- You are reminded of the necessity for ***good English and orderly presentation*** of your answers.
- ***Write algorithms and requested information in the answer booklet.***
- 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 (PL) is required to write program cod either **Standard [ISO]Pascal** or the **[ANSI] C or C#** programming language Standards may be used.
- If need be, supervisors will assist you in recording details of intermediate work carried out on the computer.
- Where information is provided as **soft copy,** notify the instructor if it is not found in your machine or has **not been made** available to you.

SECTION A: Problem Solving (Programming) **(27 marks)**

Problem: *GTHS Naah* is a highly competitive technical high school. Each term, the results of students are published in the schools notice board. It usually causes discomfort especially to students at the bottom of the class, since the result sheets contain their full identity (Class, ClassNumber, Name etc.).

A software company has been hired to provide a software solution to this problem. They will do so by implementing a program/software called "**RESULTSOFT**".

They have decided to make use of a *Cryptographic Hash function* that will generate unique Keys for every student each term. The keys are associated with their results to form a **key-value pair** and only a student can know their personal key called "**hashCode**". The hashCode is similar to an ID number because no two students will ever have the same hashCode in a term.

The hash function takes one input (N) and an arbitrary number and generates many keys.

A sample of the student database which is printed and published as result sheet for 5 students is shown below

Student Database:(Sample First Term Result Sheet)

Sn	Class	Name	ClassNumber	Average
2	F2	Natang	23	15
4	F4	Kelly	12	5
0	USS	Tambe	40	2
1	LSA	Orok	79	10
4	F3	Tim	21	7

Problem Description and Analysis

Hash Function Description

The hash function "*hash()*" takes as parameter an integer N and generates random numbers between **11** and **N+11**.

The random function in the math library of your implementation language is used for this.

A generated random number(*r*) is considered a valid ***hashCode*** if it is an odd number.

The hashCodes are stored in an array called "*hash*" as they are generated. That is, the array is initialized with unique odd numbers.

If a hashCode already exists in the array (*verify by comparing each hashCode with the hashCodes that are already in the array*), it cannot be stored again in order to avoid a collision. In this case, a different random number has to be generated and tested for validity. A Boolean variable "**exist**", initially set ***to zero***, is set **to one** if the hashCode already exists.

If the variable "**exist**" is set to ***one*** at the end of the *verification loop*, then the random number is not odd and cannot be a hashCode. We then generate another random number and test its validity. If it is set to zero, then the number is unique and can be used as a hashCode.

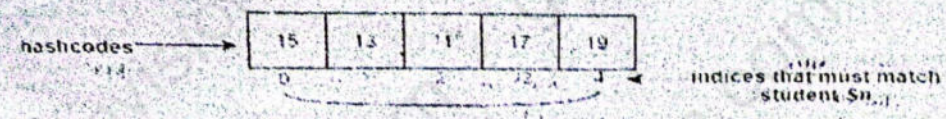
Once the array (**hash**) has *X unique numbers*, the hashCode generation process should be terminated immediately. X is the number of students whose results are to be published.

An algorithm for the hash function is given below

```
Begin
Procedure hash(int N){
Var: exist, r, counter = 0 : integer// hash[ ] & result[ ][ ] declared in global space
REPEAT
exist = 0;
r = rand()%N +11;
If r % 2 ≠ 0 Then
For i ← count – 1 to 0  do
   If r = hash[i] then
     exist = 1; break;
Endif
Endfor
   If exist = 0 then  hash[count] = r;
        count = count +1
    Endif
Endif
UNTIL count = 5;
End
```

The array now contains unique hashCodes, which will be used by each student to refer their marks.

The array will look like this:



Mapping of HashCodes

- First the hash codes are extracted from the ***hash*** array and inserted into a *new field* (***hascode***) that has been added to the student results record. The record can be regarded as a table in a database.

To do this, verify if the index in the ***array "hash"*** matches a *student's serial number* (**Sn**). If it does, assign the hashCode in that array position as the student's hash Code.
For example; the student *Natang* with serial number 2 will have the hashCode **{11}.**

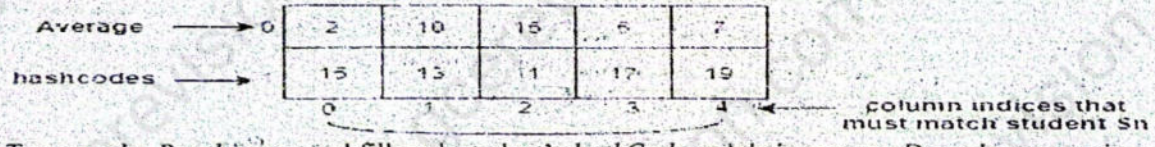
Student Database

Sn	hashcode	Class	Name	ClassNumber	Average
2	11	F2	Natang	23	15
3	17	F4	Kelly	12	5
0	15	USS	Tambe	40	2
1	13	LSA	Orok	79	10
4	19	F3	Tim	21	7

NB: The **hashcodes** in the table are just a sample, they are not what your program must generate.

- Now, copy the ***hashcode*** for each student and their average on a table (*2 dimensional array*) "***Result***" which is what will be printed and published on the Notice board. It contains only a **student's hashCode** and their*Average.*

To do so, *declare a 2-D array called Result*. The content of the array will be ***Hash codes*** *in one row* and *Averages* in another row. It will look like this:



Traverse the *Result array* and fill each student's *hashCode* and their average. Do so by comparing each student's ***serial number*** with ***column array indices***. If there is a match, fill in the students' *hashCode* and *Average* in the correct position. **Average is Integer.**

The New result sheet to be published will be like the Table (Array-"**Result**") above. Only a student whose hashCode has been given to them can know their result.

Program Implementation And Testing

Assume that there are 20 students in the school **i.e N = 20.** Consider only the sample result sheet for the program. That is **X = 5.** Thus 5 unique hashCodes are required.
NB: All variable declarations and Subroutine definitions should be in the global space.
To implement and test the reliability and efficiency of the software **(RESULTSOFT)**, the company has broken the tasks involved into the following activities that you are expected to carry out.

1. Transform the algorithm above into a Programing language Procedure ***hash(int N)*** that will generate the random numbers between *11* and *N+11* , verify if they are odd numbers and store them in the array hash[X]. **(3 marks)**
2. Define a ***record/structure*** data type "**studInfo**" to store information of X (= 5) students. **(4 marks)**
3. Declare a variable "**studResult**" of the structure type you've defined above to store information about 5 students. Only values for **average** and **hashCode** are required to run a test of the program. Initialize just this two students' information fields. Values for average should be copied from the student's database above. **(3 marks)**
4. Write a programming language procedure ***initStudInfo( )*** that allows a user to enter the values for ***Sn&Average*** for the students as in the database and in that order. **(3 marks)**
5. Write a Programming language procedure **studHash()** to copy hashCodes from the "*hash*" array and store them in the hashcode student information field. **(3 marks)**
6. Write a programming language procedure **studResultInfo(A[][])** that takes a 2D array as parameter, extracts students **hashCodes and their Averages** and stores them in the 2D array "**Result**". **(5 marks)**
7. Carry out procedure calls of the procedures defined above in the **main function** and write a block of code to print the content of the array "**result**". Write code for the main function if it is not added by default. **(4 marks)**
8. Justify whether your output matches the expected results and Print a copy of your code and output which should be attached to your answer sheet. **(2 marks)**

TURN OVER

SECTION B: Data Base Design and Implementation **(23marks)**

A garage services and repairs cars. It uses a relational database to keep track of the jobs that customers have booked for it to carry out. The database includes jobs that have been completed and jobs that are waiting to be done. The details of the jobs that the garage does, together with the parts that it stocks and uses are stored in the database using the four relations shown in the **figure below.**

Job(JobID, CarRegNo, JobDate, InGarage, JobDuration)
Car(CarRegNo, Manufacturer, Model, OwnerName, OwnerEmail, OwnerTelNo)
Part(PartID, Description, Price, QuantityInStock)
PartUsedForJob(JobID, PartID, QuantityUsed)

A partial **Data Dictionary** that shows the associated data types to different attributes is as shown below

Attribute	**Data type**
JobID, OwnerTelNo, PartID, Price, QuantityInStock, QuantityUsed	Number
InGarage	Boolean (yes/no)
Manufacturer, Model, OwnerName, OwnerEmail, CarRegNo	Char or Varchar (max-length: 20)
Description	Text
JobDate,	Date
JobDuration	Time

Your role as a database designer is to design a data model and implement a database for the Garage. Below is a catalog of activities that you must follow to design the database.

1. Draw in your answer booklet an Entity relationship diagram **(including cardinalities of relationships)** for the database schema given above. Include only the Key field(s) of each Relation. **(4 marks)**
2. You have suggested that the owner details (OwnerName, OwnerEmail, OwnerTelNo) should not be stored in the Car relation and that a new relation (**Owner**) should be created to store owner details separately from car details. The schema now has five relations (***Job, Car, Owner, Part, PartUsedForJob***).

   a. Given that *OwnerEmail* is unique, hence non key dependence, write the two relations **(Car & Owner)** and their attributes in standard notation as the other relations. Make sure Primary keys are indicated and referential integrity between the two is enforced. **(3 marks)**
   b. Explain why storing the owner details separately will improve the design of the database **(1 mark)**
3. When an appointment is made for a job, this is represented in the job relation. At the time of booking, the InGarage attribute is set to False and the JobDuration attribute is set to 0:00. When the car arrives at the garage the value of the InGarage attribute is changed to True. When the job is finished, the value of the Job duration attribute is updated to indicate how long the job took and details of the parts used are recorded in the database.
   The Job with JobID 16, of Car with registration number T20, booked on 06/01/2023 has been completed. The Job took 1hour 30 minutes (1:30) and used two of the parts with PartID 12.

a) Using a Database Management System of your choice, Create a database called **"Garage"**. Write the SQL statement that will create this database. **(2mark)**
b) *Create using SQL* statements each of the FIVE relations and POPULATE them with meaningful data of not more than 3 records in the valid domain. Write down only the SQL codes for creating the relations in your answer booklets. *Include PRINTOUTS* to show proof of the presence of data in the relations. **(8 marks)**
c) Using SQL, record in the database the fact that two of the parts with PartID 12 were used. Copy your command into your answer booklet **(2 marks)**
d) A mechanic needs to produce a list of all of the parts used on the job. The list must include PartId, Description, Price, and QuantityUsed of each part and no other details. The list should have the parts ordered by PartID in Descending Order. Implement and write the SQL command to achieve this. **(3 marks)**
e) Do a printout of the List.

End