RIVER VALLEY HIGH SCHOOL
General Certificate of Education Advanced Level
Higher 2
Preliminary Examination

COMPUTING

9569/02

Paper 2 (Lab-based)

13 Aug 2021
3 hours

Additional Materials: Electronic version of
`namelist_A.txt`
`namelist_B.txt`
`health_facilities.csv`
`cars.csv`
`customers.csv`
`rental_points.csv`
`rental_records.csv`
Insert Quick Reference Guide

READ THESE INSTRUCTIONS FIRST

Answer **all** questions.

All tasks must be done in computer laboratory. You are not allowed to bring in or take out any pieces of work or materials on paper or electronic media or in any other form.

Approved calculators are allowed.

Save each task in the given thumb drive as it is completed.

The use of built-in functions, where appropriate, is allowed for this paper unless stated otherwise.

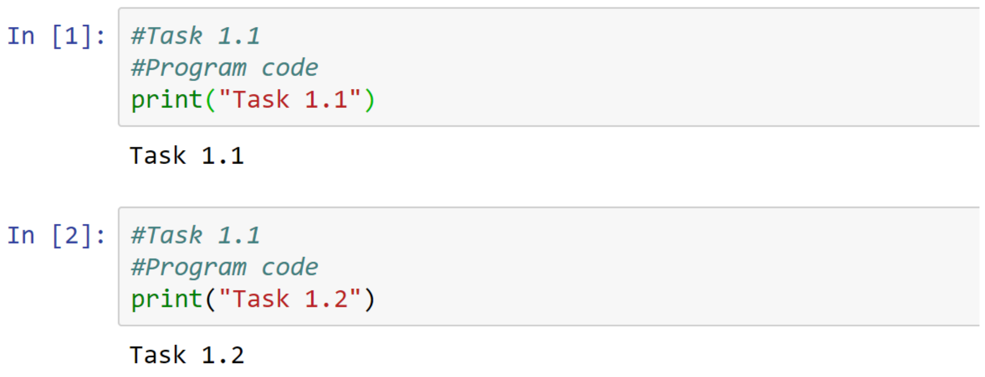
The number of marks is given in brackets [ ] at the end of each question or part question. The total number of marks for this paper is 100.

This document consists of **9** printed pages.

Instruction to candidates:

Your program code and output for each of Task 1 to 3 should be downloaded in a single `.ipynb` file.
For example, your program code and output for Task 1 should be downloaded as
`TASK1_<your name>_<centre number>_<index number>.ipynb`

1 The task is to read a single character form the keyboard, check that it is alphabetic and display the character in different number system, for example:



```
In [1]: #Task 1.1
        #Program code
        print("Task 1.1")

        Task 1.1

In [2]: #Task 1.1
        #Program code
        print("Task 1.2")

        Task 1.2
```

Task 1.1

Write a function `task1_1(name_A, name_B)` where `name_A` and `name_B` are strings which consists of only alphabet letters and spaces. The function will return `True` if

- the alphabet letters combination used in string `name_A` and `name_B` are the same and
- the spaces in string `name_A` and `name_B` are at the same locations. [7]

For example,

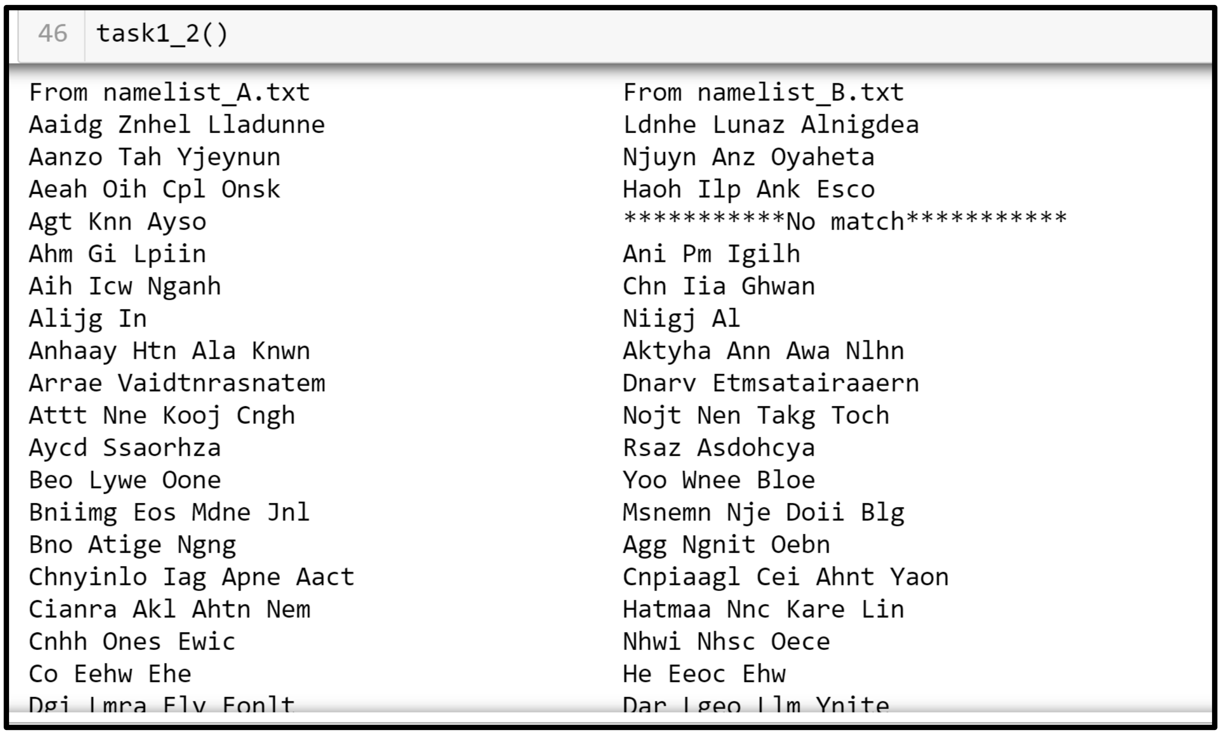
```
>>> match_names("Abcde", "Deabc")
True
>>> match_names("Abcde Fgh I", "Ihgfe Dcb A")
True
>>> match_names("Abcd Efgh I", "Ihgfe Dcb A")
False
>>> match_names("Abcde Fzh I", "Ihgfe Dcb A")
False
```

Test your program with the following test data:

```
    print(task1_1("Abcde", "Deabc"))
    print(task1_1("Abcde Fgh I", "Ihgfe Dcb A"))
    print(task1_1("Abcd Efgh I", "Ihgfe Dcb A"))
    print(task1_1("Abcde Fzh I", "Ihgfe Dcb A"))
```

Task 1.2

Write the function `task1_2()` to read the names in file "`namelist_A.txt`" and find a matching name in "`namelist_B.txt` " that satisfied the conditions stated in **Task 1.1**. If a matching name cannot be found in "`namelist_B.txt`", it will just display "`***********No match***********`". Your results should be displayed in the following manners. The time-complexity of your program code matters. [13]



```
task1_2()
```

```
From namelist_A.txt                     From namelist_B.txt
Aaidg Znhel Lladunne                    Ldnhe Lunaz Alnigdea
Aanzo Tah Yjeynun                       Njuyn Anz Oyaheta
Aeah Oih Cpl Onsk                       Haoh Ilp Ank Esco
Agt Knn Ayso                            ***********No match***********
Ahm Gi Lpiin                            Ani Pm Igilh
Aih Icw Nganh                           Chn Iia Ghwan
Alijg In                                Niigj Al
Anhaay Htn Ala Knwn                     Aktyha Ann Awa Nlhn
Arrae Vaidtnrasnatem                    Dnarv Etmsatairaaern
Attt Nne Kooj Cngh                      Nojt Nen Takg Toch
Aycd Ssaorhza                           Rsaz Asdohcya
Beo Lywe Oone                           Yoo Wnee Bloe
Bniimg Eos Mdne Jnl                     Msnemn Nje Doii Blg
Bno Atige Ngng                          Agg Ngnit Oebn
Chnyinlo Iag Apne Aact                  Cnpiaagl Cei Ahnt Yaon
Cianra Akl Ahtn Nem                     Hatmaa Nnc Kare Lin
Cnhh Ones Ewic                          Nhwi Nhsc Oece
Co Eehw Ehe                             He Eeoc Ehw
Dgi Lmra Ely Eonlt                      Dar Lgeo Llm Ynite
```

Task 2.1

Complete the doubly linked list class `Doubly_LL` by implementing both the `insert` and `delete` class functions. [10]

- The class function `insert(node)` takes a `Node` instance `node` as input and inserts it at the tail of the linked list.
  Take note that the attributes `prev` and `next` of `Node` instance `node` are both `None` before the insertion.

- The class function `delete(node)` takes a `Node` instance `node` which exists in the linked list as input. The function removes/detaches `node` from the linked list and returns it. The `node` returned has both its attributes `prev` and `next` set to `None`, but data remains unchanged.

Task 2.2

The class `LRUQ` uses the doubly linked list class `Doubly_LL` to implement its least recently used queue (`lruq`). [8]

- The attribute `hashmap` is a dictionary object that takes the node's data as key and the node instance itself in `lruq` as value.
- The attribute `size` is the max number of nodes that `lruq` can have.
- The attribute `count` is the number of nodes that `lruq` currently have.

Complete the least recently used queue class `LRUQ` by implementing the `use` function. The class function `use(value)` takes an integer `value` as input.

- If `value` is in `lruq` (referenced by hashmap), it removes the node in the `lruq` and re-insert it to the end of the `lruq`.
- If `value` is not in `lruq` (not referenced by hashmap), it references `value` in `hashmap` and inserts a new `Node` instance with `value` as its data to the end of `lruq`. If `count` > `size`, it removes the least recently used node in `lruq` and de-references it in `hashmap`.
  Hint: To de-reference a key in `hashmap`, you can call the following. `self.hashmap.pop(key, None)` where `self.hashmap` is a dictionary object.

The test function `test2_2()` is provided for you in `task2.ipynb`. The expected outcome of this test function is shown on the next page. Take note that the size of the least recently used queue is `6` in this test function.

```
Latest item used: 3
From least recently used to most recently used: Print from head:
3
----------------
Latest item used: 8
From least recently used to most recently used: Print from head:
3 8
----------------
Latest item used: 2
From least recently used to most recently used: Print from head:
3 8 2
----------------
Latest item used: 45
From least recently used to most recently used: Print from head:
3 8 2 45
----------------
Latest item used: 3
From least recently used to most recently used: Print from head:
8 2 45 3
----------------
Latest item used: 45
From least recently used to most recently used: Print from head:
8 2 3 45
----------------
Latest item used: 45
From least recently used to most recently used: Print from head:
8 2 3 45
----------------
Latest item used: 12
From least recently used to most recently used: Print from head:
8 2 3 45 12
----------------
Latest item used: 31
From least recently used to most recently used: Print from head:
8 2 3 45 12 31
```

Queue is full at this point

```
----------------
Latest item used: 42
From least recently used to most recently used: Print from head:
2 3 45 12 31 42
```

8 is removed because it is the least used

```
----------------
Latest item used: 12
From least recently used to most recently used: Print from head:
2 3 45 31 42 12
----------------
Latest item used: 12
From least recently used to most recently used: Print from head:
2 3 45 31 42 12
----------------
Latest item used: 2
From least recently used to most recently used: Print from head:
3 45 31 42 12 2
----------------
```

Task 3.1

Write program code to read the csv file "`health_facilities.csv`" and insert all information in the file as documents into a NoSQL MongoDB database called "`Health`" with one collection called "`facilities`". The "`_id`" of the documents in the database should start from `1`, `2`, `3` and `4` etc. The correct data type of each field is expected to be inserted into the database. [10]

Task 3.2

a) Write a MongoDB Pymongo query to retrieve all public acute hospital documents with their corresponding number of beds more than 7200. [4]

b) Write program code to bubble sort the results retrieved in **Task 3.2** a) according to the average number of beds per facility. Then, display the top 3 years which has the highest average number of beds per facility using the format below.

```
The three years that have the highest average number of beds per
facility are: _____, _____ and _____.
```

[7]

Task 3.3

a) Write a MongoDB Pymongo query and program code to display all "`_id`"s of Not-for-Profit health facilities documents that have no facility. [3]

b) Write MongoDB Pymongo code to update the fields "`no_of_facilities`" and "`no_beds`" of only 3 documents retrieved in **Task 3.3 a)** to `1` and a random number from `10` to `20` respectively. [6]

Task 4 – Car Loaning System

CaRent is a company providing electronic car rental services. The company engages you to design a web application using Flask microframework to aid in the car rental process.

The following information of each `Customer` is stored:
`CustomerID` – auto increment integer value to keep track of the ID of customer.
`Name` – name of customer.
`Gender` – gender of customer, to be stored as a single character, using either '`M`' or '`F`'.
`Contact` – contact number of customer.

The following information of each `Car` is stored:
`VIN` – vehicle identification number (VIN) of the car.
`Brand` – brand of the car.
`Vehicle Type` – type of the car, can be `'Sedan'`, `'Hatchback'`, `'SUV'` or `'MPV'`.
`Energy Source` – type of energy source the engine is running on, can be `'Diesel'`, `'Gasoline'`, `'Hybrid'` or `'Electricity'`.
`DailyPrice` – daily price for renting the car.
`Availability` – availability of the car, can be `'Available'` or `'Unavailable'`.

The following information of each `RentalPoint` is stored:
`PointID` – auto increment integer value to keep track of the ID of rental service point.
`Address` – address of the rental point.
`OpWeekDay` – weekdays that the rental point is open, stored as a 7-digits string, starting from Sunday to Saturday, with `'1'` indicating open and `'0'` indicating closed. E.g. `'0111110'` means it is open on weekdays and closed on weekend.
`OpStartHr` – starting time of daily operation, stored as a 4 digits string, using 24hour time format.
`OpEndHr` – ending time of daily operation, stored as a 4 digits string, using 24hour time format.

The following information of each `RentalRecord` is stored:
`CustomerID` – ID of customer.
`VIN` – VIN of car.
`StartDate` – start date for the rental service.
`CollectionPointID` – ID of the collection point.
`ReturnDate` – return date for the rental service.
`ReturnPointID` – ID of the return point.

The information is to be stored in four tables:
`Customer`
`Car`
`RentalPoint`
`RentalRecord`

Task 4.1

Create an SQL file called `Task4_1.sql` to show the SQL code to create the database `car_rental.db` with the three tables.

The table `Customer` must use `CustomerID` as its primary key, the table `Car` must use `VIN` as its primary key, and the table `RentalPoint` must use `PointID` as its primary key.

The table `RentalRecord` should use `CustomerID`, `VIN` and `StartDate` as a composite key, while `CustomerID`, `VIN` and `CollectionPointID`/`ReturnPointID` must refer to `CustomerID` in `Customer`, `VIN` in `Car` and `PointID` in `RentalPoint` as foreign keys.

Save your SQL code as
`Task4_1.sql`

[6]

Task 4.2

The files `customers.csv`, `cars.csv`, `rental_points.csv` and `rental_records.csv` contains information about the customers, cars, rental points and the past rental records. The first row of each file contains the header of the respective columns. Each row in the files is a comma-separated list of information.

Write a Python program to insert all information from the three files into the database `car_rental.db`. Run the program.

Save your program code as
`Task4_2.py` [6]

Task 4.3

You are tasked to implement a function to search and display all past rental records of a customer. Using the customer's name **`'Goh Yi Xi'`**, query and display a list of data with the following fields as shown in the table, sorted in the ascending order according to the start date.

Name	Contact	VehicleType	StartDate	ReturnDate	DailyPrice
…	…	…	…	…	…

Write the SQL code required.

Save this code as
`Task4_3.sql` [5]

Task 4.4

The company wants to implement a function to register new cars for rental into the database. Office staff can register new cars by adding the values of the attributes in the `Car` table.

Write a Python program and the necessary files to create a web application that:

- Receive the following information:
  - `VIN`, `Brand`, `VehicleType`, `EnergySource`, and `DailyPrice` of a car through a HTML form.
  - `Availability` should be set to the default value of `'Available'`.
  - Note that `VehicleType` and `EnergySource` should be in **dropdown** list format to improve data validity.

- Check if the `VIN` is valid based on the following algorithm:
  - Step 1: Translate all letters to integer values using the following table (`I`, `O`, and `Q` are not allowed in a valid `VIN`):

A: 1	**B**: 2	**C**: 3	**D**: 4	**E**: 5	**F**: 6	**G**: 7	**H**: 8	**N/A**
J: 1	**K**: 2	**L**: 3	**M**: 4	**N**: 5	**N/A**	**P**: 7	**N/A**	**R**: 9
N/A	**S**: 2	**T**: 3	**U**: 4	**V**: 5	**W**: 6	**X**: 7	**Y**: 8	**Z**: 9

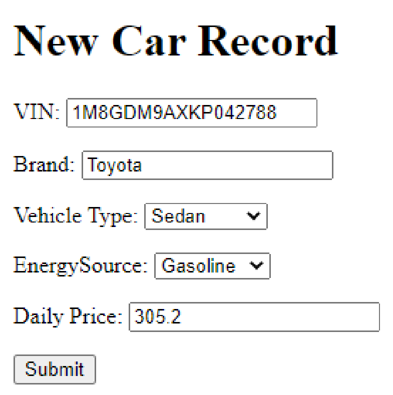
  - Step 2: Use the following weight factor for each position in the VIN. **The 9th position is that of the check digit.** Its weight factor has been substituted with a `0`, which will cancel it out in the multiplication step.

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Weight	8	7	6	5	4	3	2	10	0	9	8	7	6	5	4	3	2

  - The sum of product of the letter/digit with their corresponding weight factor is then divided by `11`.
  - The remainder is the check digit. If the remainder is `10`, the check digit will use `X` instead.
  - E.g. the `VIN` with values `1M8GDM9A_KP042788` will produce a check digit of `X` and hence `1M8GDM9AXKP042788` is a valid `VIN`.

- If `VIN` is valid, create a new car record in the `Car` table, and display the record in the confirmation page.
- Otherwise, inform the user that the `VIN` is invalid.

Input Page



New Car Record

VIN: 1M8GDM9AXKP042788

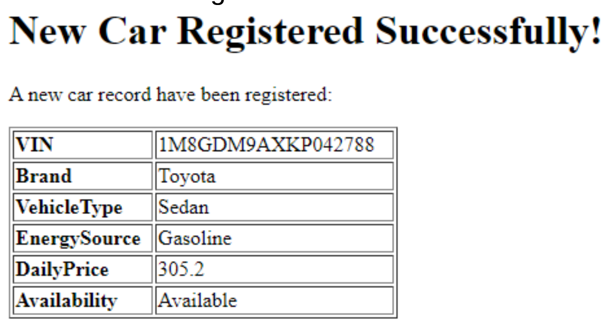
Brand: Toyota

Vehicle Type: Sedan

EnergySource: Gasoline

Daily Price: 305.2

Valid VIN and register successful:

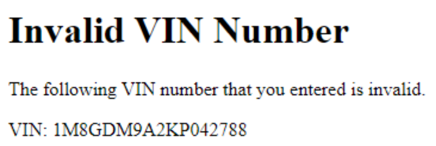


New Car Registered Successfully!

A new car record have been registered:

VIN	1M8GDM9AXKP042788
Brand	Toyota
VehicleType	Sedan
EnergySource	Gasoline
DailyPrice	305.2
Availability	Available

Invalid VIN:



Invalid VIN Number

The following VIN number that you entered is invalid.

VIN: 1M8GDM9A2KP042788

You may assume:
- All inputs are in valid format.
- VIN: 1M8GDM9AXKP042788 is a new record to the database

Save your program as
`Task4_4.py`
With additional files or sub-folders as needed in a folder named
`Task4_4`

Run the web application. Enter the values based on the sample input above.
Then save the output of the program as `Task4_4.html`. [15]