1 [ACJC/PRELIM/9569/2021/P1/Q1]

A famous restaurant only accommodates one seating daily from 6.00 pm to 8.00 pm. It has 10 tables, each with a maximum capacity between 2 and 8 people. Advanced reservation is required to dine in at the restaurant.

The owner of the restaurant decides to write a program to handle reservations. As a trial, it can only take a booking for one evening only.

A procedure to initialise the arrays `MaxSize`, `IsBooked` and `GroupSize` has been defined. The indexes of each array corresponds to the table number.

Index	`MaxSize`		`IsBooked`		`GroupSize`
1	2	1	FALSE	1	
2	2	2	FALSE	2	
3	4	3	FALSE	3	
4	4	4	FALSE	4	
5	4	5	FALSE	5	
6	6	6	FALSE	6	
7	6	7	FALSE	7	
8	6	8	FALSE	8	
9	8	9	FALSE	9	
10	8	10	FALSE	10	

The procedure `BookTable` is shown below. When a booking enquiry is made, the number of customers is keyed in.

```
01 PROCEDURE BookTable
02     DECLARE NumberOfCustomers, TableNumber :  INTEGERS
03     DECLARE Found :  BOOLEAN
04     INPUT NumberOfCustomers
05     TableNumber ← 0
06     FOUND ← False
07     REPEAT
08         TableNumber ← TableNumber + 1
09         IF MaxSize[TableNumber] > NumberOfCustomers AND
IsBooked[TableNumber] = FALSE
10             THEN
11                 Found ← TRUE
12         ENDIF
13     UNTIL Found = TRUE AND TableNumber = 10
14         IF Found = FALSE
15             THEN
16             OUTPUT "No tables with enough seats available."
17             ELSE
18                 GroupSize[TableNumber] ← NumberOfCustomers
19                 OUTPUT "Booking is successful!  Table no:", TableNumber
20     ENDIF
21 ENDPROCEDURE
```

(a) There are two errors and one missing line of code in the procedure above.

(i) Name the type of the errors. [1]

(ii) Describe the errors and the changes required to correct them. [3]

(iii) Write the missing line of code and state where it should be located. [2]

(b) Once the procedure `BookTable` is able to run correctly, the owner decides to improve its functionality.

The procedure should ask the user to input the name and the mobile number of the person making the reservation when a table with enough seats can be found.

Name and describe two data validation techniques that can be applied to any of the inputs mentioned above. [2]

(c) Explain the difference in the type of memory allocation for an array and a linked list. [2]

2 [ACJC/PRELIM/9569/2021/P1/Q2]

A hash table has 8 spaces to store strings, indexed from 1 to 8 inclusive.

The hash function finds the ASCII number of the first letter of the string, then counts the number of 1s in its binary representation. This is the index in which the string will be inserted into the hash table.

For example, the string `'Arlington'` will have index 2 because the ASCII number of `'A'` is 65, which is `1000001` in binary, and there are two `1`s.

The following strings are to be inserted into the hash table in the order given.

```
'Grover',
Horsburgh',
'Island',
'Jordan',
'Kalman'
```

(a) Find the output of the hash function for each of the strings. [5]

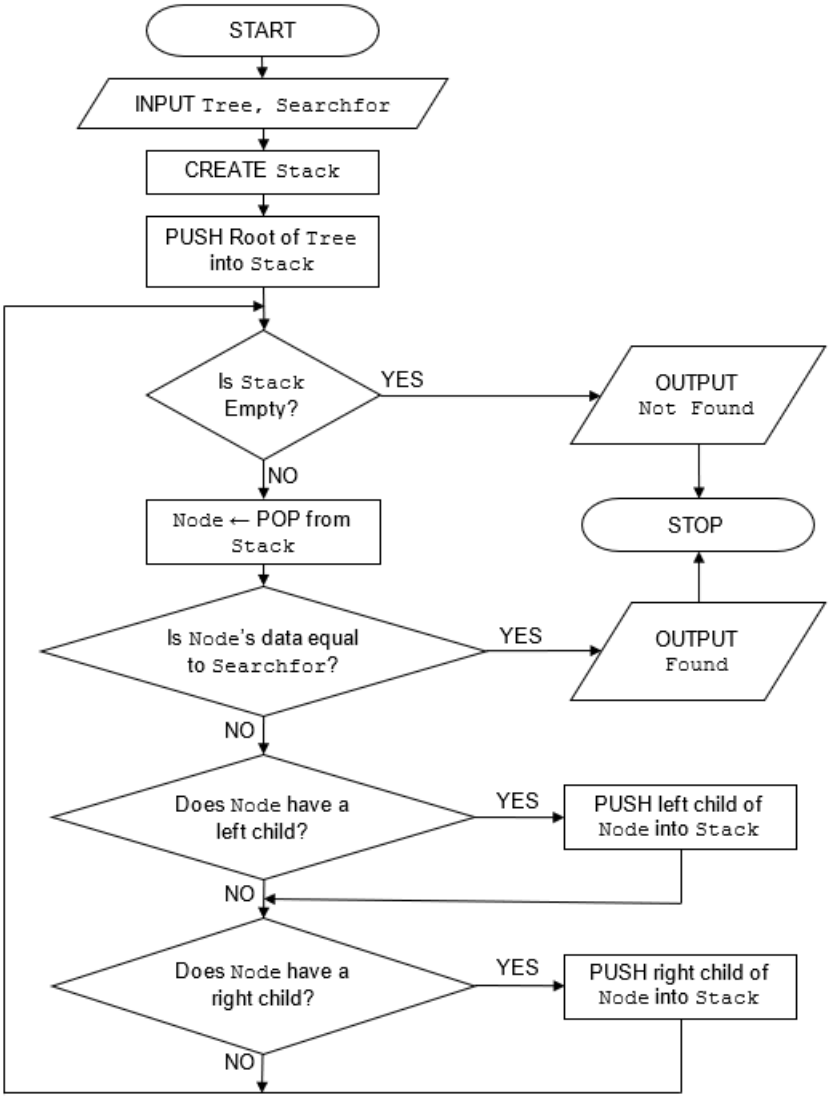
(b) **(i)** Suppose collisions in the hash table are to be resolved using open hashing.
Draw the hash table after all five strings are inserted. [5]

(ii) Suppose instead that collisions in the hash table are to be resolved using closed hashing, where spaces 6 to 8 (inclusive) are used as the overflow storage.
Draw the hash table after all five strings are inserted. [2]

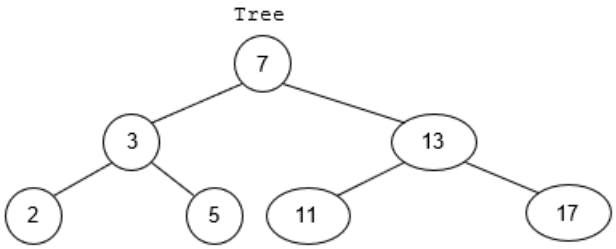
(c) Explain why the space with index 1 in the hash table will never be occupied unless there is a collision. [2]

3 [ACJC/PRELIM/9569/2021/P1/Q3]

The diagram below shows a flowchart for performing a search through a binary tree. The algorithm searches through `Tree` for `Searchfor`. If it finds a node whose data is equal to `Searchfor`, it outputs `Found`. Otherwise, it outputs `Not Found`.



(a) Given the input `Tree` below and a `Searchfor` value of 5, draw a trace table to illustrate the algorithm. [5]



(b) State whether this is a depth-first search or a breadth-first search. [1]

(c) Draw a flowchart to illustrate how the other kind of search in (ii) can be carried out using the same input parameters. [3]

(d) Given the same input `Tree` and the same `Searchfo`r value of 5, draw a trace table to illustrate the algorithm in (iii). [5]

4 [ACJC/PRELIM/9569/2021/P1/Q4]

Merge sort and bubble sort are two sorting algorithms that can be applied to sort a list of integers in ascending order.

(a) By briefly comparing the operation of merge sort and bubble sort, state which algorithm would be more efficient. [3]

The pseudocode for the recursive `MergeSortDesc` function is shown below, which takes in a list of integers and returns a new list with the integers sorted in descending order. It makes use of the `MergeDesc` function in line `10` that merges two lists of integers sorted in descending order into a single list of integers sorted in descending order as well.

```
01 FUNCTION MergeSortDesc(MyList:  LIST) RETURNS LIST
02     MaxIndex ← LENGTH(MyList)
03         IF ......A......
04             THEN
05             Half ← ......B......
06             LeftList ← LEFT(MyList, Half)
07             RightList ← RIGHT(MyList, Half)
08             SortedLeftList ← MergeSortDesc(LeftList)
09             SortedRightList ← MergeSortDesc(RightList)
10             Result ← MergeDesc(SortedLeftList, SortedRightList)
11         ELSE
12             ......C......
13         ENDIF
14         RETURN Result
15 ENDFUNCTION
```

(b) State what is meant by a **recursive** function. [2]

(c) Write the pseudocode for `A`, `B` and `C` in the algorithm. [3]

(d) Describe the operation of the `MergeDesc` function.

Assume that the function does not modify the two input lists. [4]

5 [ACJC/PRELIM/9569/2021/P1/Q5]

A gym organises various classes and runs a loyalty membership programme with four tiers: Bronze, Silver, Gold and Diamond

Upon joining, each member is given a unique membership number and starts with a Bronze membership. Each member can sign up for multiple classes at a reduced rate based on the membership tier.

Each class has a unique class name. Some classes are offered at three different levels: Beginner, Intermediate and Advanced

Each instructor is identified with a unique three-character code and can take one or more classes.

A relational database is to be created to store data about members, employees and classes.

Part of the table `MEMBER`, which is a first attempt at the database design, is shown below.

MemberNo	MemberName	MemberTier	ClassName	InstCode
3*5	3*Lindy White	3*Silver	Body Pump	WAY
			Yoga (Beginner)	DAV
			Zumba	ROG
...	...	...	...	...
78	Derek Davis	Bronze	Muay Thai (Beginner)	CHA
...	...	... ...	...	...
4*132	4*John Chua	4*Diamond	Circuits (Intermediate)	JON
			Muay Thai (Intermediate)	LEX
			Yoga (Advanced)	DAV
			Zumba	ROG
...	...	...	...	...

(a) The table `MEMBER` is not normalised.

(i) Describe **one** potential issue that may be encountered when the data are maintained in such a non-normalised table. [1]

(ii) Explain why the table is not in first normal form (1NF). [1]

(b) A second attempt at the database design gives rise to two tables:

```
MEMBER(MemberNo, MemberName, MemberTier)

MEMBERCLASSES(MemberNo, ClassName, Instructor)
```

The primary keys are not shown.

(i) State what is meant by a **primary key**. [1]

(ii) By referring to the relationship between the tables `MEMBER` and `MEMBERCLASSES`, state how the relationship is implemented. [2]

(iii) Write an SQL query to create the table MEMBER with the appropriate constraints. [4]

(c) Another attempt at the database design needs to be made to ensure that all the tables are in third normal form (3NF).

In addition, the following data need to be recorded in the database:

- the date on which each member signs up for the gym membership;
- the attendance of each member in any classes taken;
- the original fee, i.e. before discount, of each class;
- the name and the salary of each instructor.

(i) State the total number of 3NF tables required and give their names. [1]

(ii) Draw the Entity-Relationship (E-R) diagram to show the 3NF tables and the relationships between them. [4]

(iii) A table description can be written as:

```
TableName(Attribute1, Attribute2*, Attribute3, ...)
```

The primary key is indicated by underlining one or more attributes. Foreign keys are indicated by using a dashed underline/asterisk.

Using the information provided, write table descriptions for all the 3NF tables you identified in **(c)(i)**. [8]

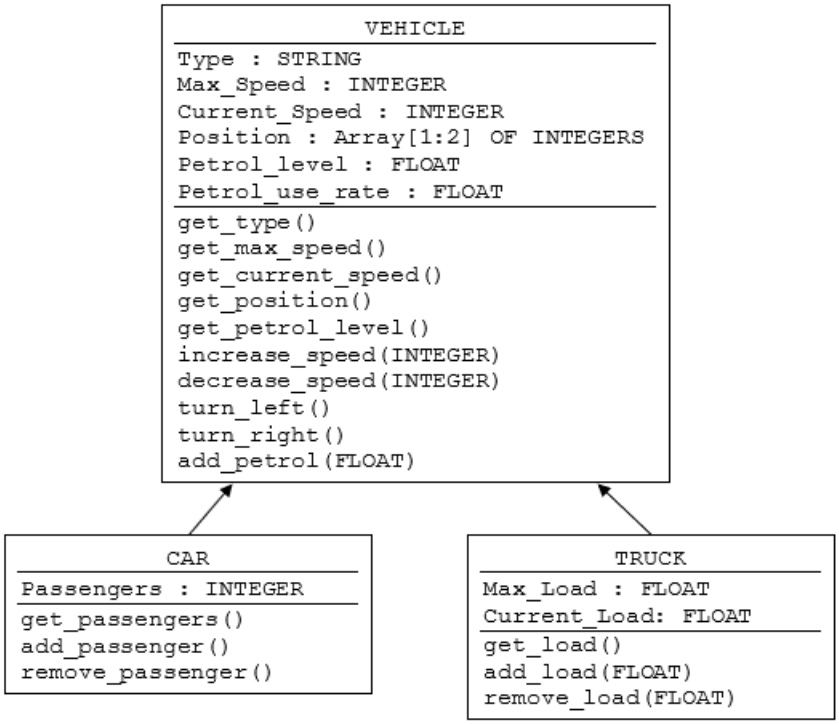
(d) Making backups and archives are performed to prevent the loss of data.

Explain the difference between a backup and an archive. [2]

6 [ACJC/PRELIM/9569/2021/P1/Q6]

A driving simulator is programmed using Object-Oriented Programming (OOP).

The diagram below shows a UML Class Diagram with **some** of the classes, attributes, and methods used in the simulator.



(a) State the relationship between the `CAR` class and the `VEHICLE` class. [1]

(b) Explain briefly, in this context, how each of the following features of Object-Oriented Programming help the simulation to be developed more efficiently.

(i) Abstraction [2]

(ii) Inheritance [2]

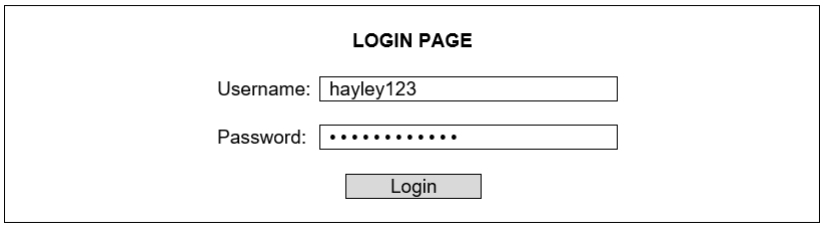
(c) The petrol use rate depends on the speed at which the vehicle is travelling, as well as the mass of the vehicle and the contents of the vehicle – the number of passengers in a car, or the mass of the load in a truck. Explain how polymorphism can be used in this case to write the simulation. [2]

7 [ACJC/PRELIM/9569/2021/P1/Q7]

A new social media platform is to be created. In years to come, it is expected to be as popular globally as other trending social media platforms.

(a) Give two reasons why a NoSQL database is likely to be more suitable than an SQL database for the social media platform. [2]

A basic login page that controls access to user accounts is shown below. The password field masks the user input with a dot (•) replacing each of the characters supplied.



(b) When the login button is clicked, the program processes the username and password supplied by the user. It displays an error message if the username entered does not exist in the database. If the password entered matches the registered password for the username, login is granted. Otherwise, the program displays an error message to indicate the user of the incorrect password entered.

The account will be locked if the user enters the correct username, but enters the wrong password three times.

(i) Create a decision table to show these conditions and actions. [4]

(ii) Simplify your decision table by removing redundancies. [1]

(c) It is known that users tend to have different problems associated with passwords.

Besides the error message to tell the user when an incorrect password is entered, describe **two** examples based on usability principles that can be applied to improve the functionality of the login page. [2]

(d) Explain why the HTTP POST method should be used instead of the HTTP GET method for the login request. [2]

8 [ACJC/PRELIM/9569/2021/P1/Q8]

In a hypothetical scenario, a data security company is helping a client company manage a database of the client company's customers. The data security company notices a possible vulnerability in the database.

Further investigation shows that the vulnerability is obscure and that none or few of the programmers anticipated it. Since the vulnerability is obscure, they determine that the chances of the database being breached are minimal, and decide not to tell the client company about it.

Instead, the data security company waits until the next time the database is due for scheduled maintenance to attempt to fix the vulnerability. By doing so, they can give themselves enough time to learn how to fix the vulnerability and avoid causing unnecessary panic within the client company or among the customers, which could lead to a potential loss of business.

Describe how each of the following ethical guidelines was breached by the data security company.

(a) Integrity [2]

(b) Responsibility [2]

(c) Competence [2]

(d) Professionalism [2]

9 [ACJC/PRELIM/9569/2021/P2/Q1]

The Universal Product Code (UPC) system is used for tracking trade items in shipping, inventory, and sales. Each item is given a 12-digit identification number. The validity of this identification number can be checked using a checksum. If x_i represents the ith digit (starting with $i = 1$ as the leftmost digit), then a valid identification number satisfies the condition that

$$3x_1 + x_2 + 3x_3 + x_4 + 3x_5 + x_6 + 3x_7 + x_8 + 3x_9 + x_{10} + 3x_{11} + x_{12}$$

has a remainder of 0 when divided by 10.

The identification number can be encoded into a barcode. For this Task, the barcode will be represented as a string of '0's and '1's, where '0' represents a white stripe and '1' represents a black stripe.

The barcode is divided into seven sections. From left to right, they are

- A 'quiet zone' consisting of nine '0's;
- A start pattern which is always '101';
- The first six digits of the identification number are encoded using the table below;
- A middle pattern which is always '01010';
- The last six digits of the identification number are encoded using the table below;
- An end pattern which is always '101';
- A 'quiet zone' consisting of nine '0's.

The table below shows the encoding system for the digits. Note that depending on whether the digit occurs in the first six digits or the last six digits, it would be encoded differently. However, the two encodings are optical inverses of each other – a '0' is changed into a '1', and vice versa.

Digit	First six digits	Last six digits
0	'0001101'	'1110010'
1	'0011001'	'1100110'
2	'0010011'	'1101100'
3	'0111101'	'1000010'
4	'0100011'	'1011100'
5	'0110001'	'1001110'
6	'0101111'	'1010000'
7	'0111011'	'1000100'
8	'0110111'	'1001000'
9	'0001011'	'1110100'

The reason for encoding the first and last six digits differently is that the barcode may inadvertently be scanned upside down. Notice that in the first six digits, the encoding for each digit contains an odd number of '1's, while in the last six digits, the encoding for each digit contains an even number of '1's. This allows the scanning software to detect if the barcode has been placed upside down and correct it.

For example, the UPC identification number 036000 291452 would be encoded as:

000000000	101	0001101	0111101	0101111	0001101
Quiet	Start	0	3	6	0

0001101	0001101	01010	1101100	1110100	1100110
0	0	Middle	2	9	1

1011100	1001110	1101100	101	000000000
4	5	2	End	Quiet

(Notice that in an actual barcode, the stripes for the start, middle and end pattern are usually slightly longer than the surrounding stripes. This is to help humans to read it.)

Task 1.1

Write a function to determine the validity of any input string as an identification number. [5]

Task 1.2

Write a function to convert a valid identification number, given as a string, into a barcode (a string of '0's and '1's). [5]

Task 1.3

Write a function that takes in a string, check whether it represents a valid barcode, and converts it to an identification number if it does. Note that the barcode may be upside down. [11]

Download your program code and output for Task 1 as `TASK1_<your name>_<centre number>_<index number>.ipynb`

10 [ACJC/PRELIM/9569/2021/P2/Q2]

A file compression algorithm reduces file sizes so that files can be sent more quickly. One such algorithm is the Huffman algorithm for text files, which will be implemented in this task.

Unlike ASCII, which assigns a fixed size of 8 bits for each character, the Huffman algorithm assigns fewer bits to more common characters and more bits to less common characters. For example, in a long text written in English, characters such as '`e`' and '`t`' will have fewer bits assigned to them than characters such as '`q`' and '`z`'. If the text is long enough, this will use fewer bits in total to encode the text compared to ASCII.

To know which sequence of bits to encode for each character, the **frequency** of each character, which is the number of times each character appears in the text file, is tabulated.

The characters are put into a tree. A node is created for each character. The following steps are then repeated until there is only one node without a parent:

(a) Identify the two nodes, without parents, which have the lowest frequency.

(b) Create a new node whose left and right children are the two nodes identified in Step 1. The frequency of the new node is the total of the frequency of its children.

The diagram on the following page shows the process of creation of a tree for a file with only five distinct characters ('`A`', '`E`', '`I`', '`O`' and '`U`'), in five stages.

The bit sequence assigned to a character will be the path from the root to the node corresponding to that character, where going left corresponds to '`0`' and going right corresponds to '`1`'. For example, '`A`' is encoded as '`10`' and '`O`' is encoded as '`011`'.

Task 2.1

Create a Node class that has the following attributes:

- `data`, which is determined when the node is initialized
- `left`, a pointer to another node,
- `right`, a pointer to another node

When the node is initialised, left and right do not point to anything. The class also has setter methods for left and right, and getter methods for all three attributes. [3]

Task 2.2

Write code that takes an input `.txt` file and creates a dictionary whose keys are the characters in the file, including spaces, punctuation and line breaks ('\n'), and the value of a key is its frequency in the file. Uppercase and lowercase letters should be considered as different characters.

Create a node for each character in the file, and put the nodes into a list in ascending order of frequency. [11]

Task 2.3

Create a tree using the algorithm described above. [5]

Task 2.4

Create a dictionary whose keys are the characters, and the value of a key is the bit sequence of that character, expressed as a string of '0's and '1's.

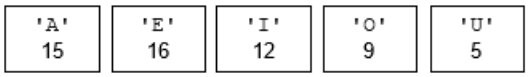
Carry out Tasks 2.2 to 2.4 on the file `HAMLET.txt`. Compress the file by replacing each character with its bit sequence and writing the output to a new file, `HAMLET_compressed.txt`. [8]

Download your program code and output for Task 2 as `TASK2_<your name>_<centre number>_<index number>.ipynb`

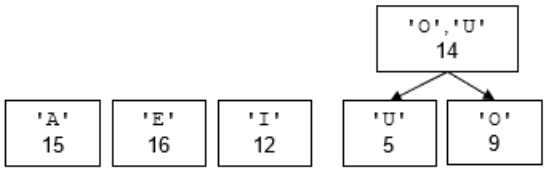
Diagram showing how the tree is created based on the frequency of each character:

Character	Frequency
'A'	15
'E'	16
'I'	12
'O'	9
'U'	5

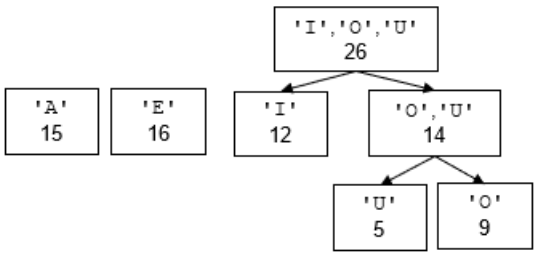
1)



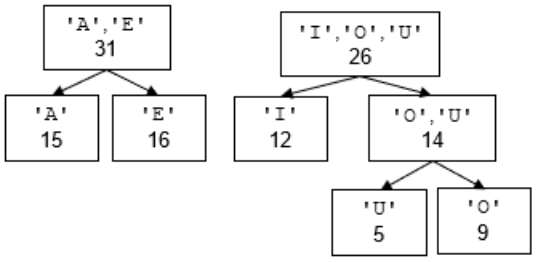
2)



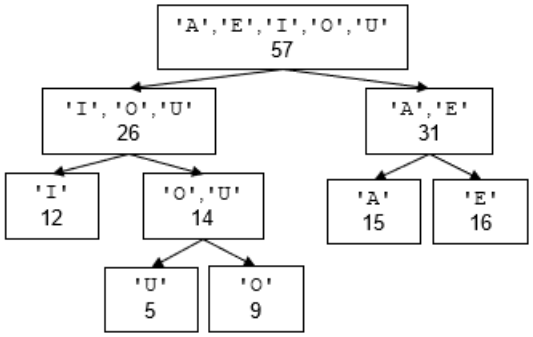
3)



4)



5)



11 [ACJC/PRELIM/9569/2021/P2/Q3]

A community centre is required to keep COVID-19 vaccination records of its members in a NoSQL database. The CoviDie vaccine, which requires two doses to be taken at least 21 days apart, has been secured for all members of this particular community centre.

As everyone needs to be vaccinated before the end of 2021, we shall only consider the year 2021, which is a non-leap year. The table below shows the number of days available in the twelve months of 2021.

Month	01	02	03	04	05	06
Days	31	28	31	30	31	30

Month	07	08	09	10	11	12
Days	31	31	30	31	30	31

Task 3.1

Write a function `second_dose_date(date)` that:

- Takes a string value `date` in the format `YYYYMMDD`, where `YYYY` represents the year, `MM` represents the month and `DD` represents the day
- determines the date that is 21 days after the input date
- returns the result date in the format `YYYYMMDD`

Assume that the result date does not go beyond `20211231`. [3]

Test the function using the following three calls:

- `second_dose_date('20210105')`
- `second_dose_date('20210212')`
- `second_dose_date('20210919')` [3]

Save your program code as

`TASK3_1_<your name>_<centre number>_<index number>.py`

Task 3.2

The list of members under the management committee of the community centre is stored in the text file `VACCINATION.txt`. Some members have not taken the vaccination at all, some others have only taken the first dose, while the rest have taken both doses.

Each line of the text file is of the following format: `_id,name,date_first_dose,date_second_dose,remarks`

- `_id` is a unique integer ID assigned to the member
- `name` is the name of the member
- `date_first_dose` and `date_second_dose`, if any, represent the date of the first dose and the date of the second dose respectively in the format `YYYYMMDD`
- `remarks`, if any, shows the pre-existing condition of the member

Write program code to insert the data from `VACCINATION.txt` into a NoSQL database `community_centre` under the collection `management_committee`. The program should clear the collection `management_committee` if it exists inside the database. [7]

Save your program code as `TASK3_2_<your name>_<centre number>_<index number>.py`

Task 3.3

The community centre needs a program to check the vaccination status of its members. The program should also allow for the downloading of vaccination certificates for members who are fully vaccinated, i.e. they have taken the two doses.

Write program code to:

- prompt the user to input a member ID, and keep prompting until the user keys in numeric character(s)
- if the member ID is available in the NoSQL database, perform either one of the following:
  - if the member is fully vaccinated, write the vaccination certificate to an output text file and update the record in the NoSQL database by including a field and an appropriate value to indicate that the certificate has been downloaded
  - if the member has only taken the first dose, output a message to show the date from which the member can take the second dose

  - if the member has not taken the vaccination at all, output a message to tell that the member should take the first dose as soon as possible

- otherwise, if the member ID is not available in the NoSQL database, display an appropriate message and terminate the program

The format of the vaccination certificate is as follows.

```
VACCINATION CERTIFICATE

Name:  <name>

Vaccine type:  CoviDie

Date of first dose:  <date_first_dose>

Date of second dose:  <date_second_dose>
```

[8]

Save your program code as `TASK3_3_<your name>_<centre number>_<index number>.py`

Test your program for the member with `member_id = 24`. [2]

The output text file should be saved as `TASK3_3_<your name>_<centre number>_<index number>.txt`

12 [ACJC/PRELIM/9569/2021/P2/Q4]

A company specialising in bento boxes wishes to trial a relational database management system to manage its data. It is expected that the database should be normalised to third normal form (3NF).

The company owns four kiosks. For each of the kiosks, the following information is to be recorded in the table `Kiosk`:

- `KioskID` – the unique integer assigned to the kiosk
- `Location` – the area where the kiosk is located
- `Rating` – the average rating of the kiosk between 0.0 and 5.0 inclusive

The company offers eight different types of bento boxes. Some of them may contain egg, nut, seafood or a combination of them. For each of the bento boxes, the following information is to be recorded in the table `BentoBox`:

- `BentoName` – the unique name of the bento box
- `ProductionCost` – the cost incurred in producing the bento box in dollars and cents
- `ContainEgg` – an integer 0 for not containing egg and 1 for containing egg
- `ContainNut` – an integer 0 for not containing nut and 1 for containing nut
- `ContainSeafood` – an integer 0 for not containing seafood and 1 for containing seafood

Each of the four kiosks sells all eight bento boxes at different mark-up prices. Another table `KioskBento` is needed to record the following information:

- `KioskID` – the unique integer assigned to the kiosk
- `BentoName` – the unique name of the bento box
- `SellPrice` – the price at which the bento box is sold at the kiosk in dollars and cents

Task 4.1

Create an SQL file called `TASK4_1_<your name>_<centre number>_<index number>.sql` to show the SQL code to create the database `bento_company.db` with the three tables. [5]

Save your SQL code as `TASK4_1_<your name>_<centre number>_<index number>.sql`

Task 4.2

The files `KIOSK.txt` and `BENTOBOX.txt` contain information about the company's kiosks and bento boxes respectively for insertion into the database. Each row in the two files is a comma-separated list of information.

For `KIOSK.txt`, information about each kiosk is given in the following order: `KioskID`, `location`, `rating`

For `BENTOBOX.txt`, information about each bento box is given in the following order: `BentoName`, `ProductionCost`, `ContainEgg`, `ContainNut`, `ContainSeafood`

The mark-up price for each kiosk has been set as follows:

- `KioskID = 1` sells each bento box at a price that is $2.60 higher than the production cost
- `KioskID = 2` sells each bento box at a price that is $2.90 higher than the production cost
- `KioskID = 3` sells each bento box at a price that is $2.40 higher than the production cost
- `KioskID = 4` sells each bento box at a price that is $3.10 higher than the production cost

Write program code to insert all the required information into the database `bento_company.db`. [6]

Save your program code as `TASK4_2_<your name>_<centre number>_<index number>.py`

Run your program.

Task 4.3

The company wishes to create a form to display the bento boxes sold at a particular kiosk and their prices in a web browser. The form should allow customers to indicate egg, nut and seafood allergies, if any, and filter out the bento boxes that they cannot consume.

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

- receives input from a HTML form that includes:
  - a text box to enter the `location` of the kiosk
  - three checkboxes to indicate egg, nut and seafood allergies, if any
- returns a HTML document to display only the bento boxes that the customers can consume based on the allergies indicated, if any, and their prices for the given `location`

Input validation is not required. [10]

Save your Python program as `TASK4_3_<your name>_<centre number>_<index number>.py`

with any additional files / sub-folders as needed in a folder named `TASK4_3_<your name>_<centre number>_<index number>`

Run and test the web application using the following input:

- `'Woodlands'` entered as the `location`
- checkboxes indicating egg and seafood allergies ticked [2]

Save the output of the program as `TASK4_3_<your name>_<centre number>_<index number>.html`