ANGLO-CHINESE JUNIOR COLLEGE
JC2 PRELIMINARY EXAMINATION

Higher 2

COMPUTING

9569/02

Paper 2 (Lab-based)

12 August 2021

3 hours

Additional Materials: Electronic version of `HAMLET.txt` data file
Electronic version of `VACCINATION.txt` data file
Electronic version of `KIOSK.txt` data file
Electronic version of `BENTOBOX.txt` data file
Insert Quick Reference Guide

READ THESE INSTRUCTIONS FIRST

Answer **all** questions.

All tasks must be done in the 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 as it is completed.

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

Note that up to **6** marks out of 100 will be awarded for the use of common coding standards for programming style.

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 and **1** blank page.

Instruction to candidates:

Your program code and output for each of Task 1 and 2 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 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 i^{th} 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`

2 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:

1. Identify the two nodes, without parents, which have the lowest frequency.
2. 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

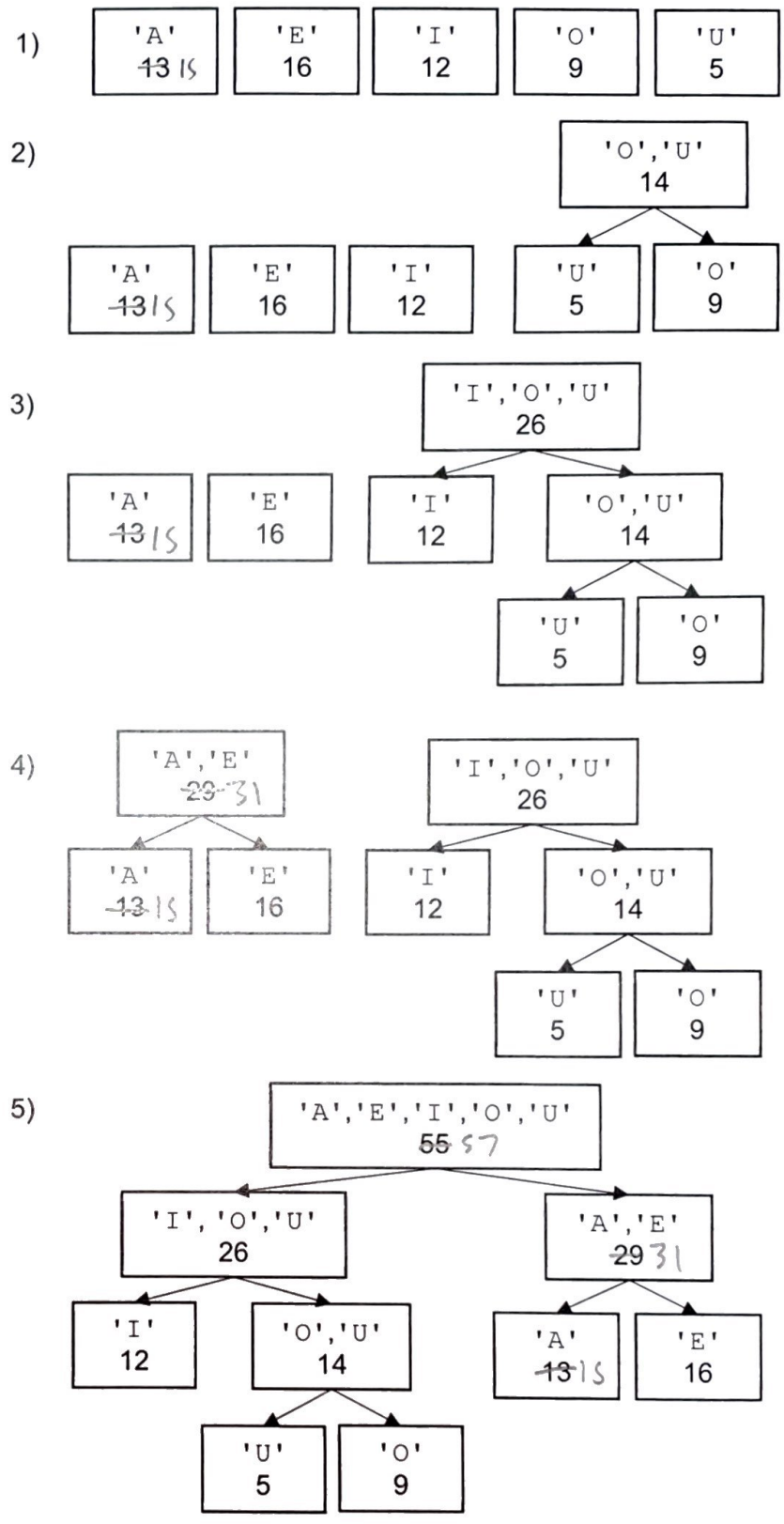
Create a dictionary whose keys are the characters, and the value of a key is the bit sequence of that character, expressed as a string of `'0'`s and `'1'`s.

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'	~~13~~ 15
'E'	16
'I'	12
'O'	9
'U'	5



3 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`

4 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`