



COMPUTING

9569/02

Paper 2 (Lab-based)

28 August 2024 (Wednesday)

3 hours

Additional Materials: Electronic version of votes.txt data file (Task 1)
 Electronic version of scores.txt data file (Task 2)
 Electronic version of customers.txt, products.txt, orders.txt and
 reviews.json data files (Task 3)
 Electronic version of .jpg and .png image files (Task 4)
 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. You are allowed to use Windows Calculator.

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.

There are two folders in your network drive: **RESOURCES**: data files, **OUTPUT**: your submissions.
Near the end of the exam, you should deposit **ONLY** the files that you are submitting to the network drive OUTPUT folder. Files not in this OUTPUT folder may not be marked.

Your program code and output for **each** of Task 1 to 4 should be saved in a single .ipynb file using Jupyter Notebook, unless otherwise stated. For example, your program code and output for Task 1 should be saved as:

TASK1_<your name>_<centre number>_<index number>.ipynb

Make sure that each of your .ipynb files shows the required output in Jupyter Notebook.

This document consists of **8** printed pages and **0** blank page.

1 Name your Jupyter Notebook as:

TASK1_<your name>_<centre number>_<index number>.ipynb

For each of the sub-tasks, add a comment statement at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to, for example:

In [1]: *#Task 1.1*

Program code

Output:

A local media corporation is hosting a Top TV Artists Award. People can vote online for their favourite artist, out of 20 shortlisted ones. The voting results are stored in a file named `votes.txt`, where each line contains the voter's moniker followed by the artist they voted for, separated by a space. The following shows the first few records:

```
happypapaya Kieran_Tay
avgstudent Alarice_Soh
candycadet Amiyah_Kapoor
```

Task 1.1

Write program code to read the data from `votes.txt` and store the number of votes each artist received in a 2D list `results`. Each row should contain the artist's name and the corresponding vote count. [4]

Task 1.2

Write program code that applies the insertion sort to the 2D list `results`, sorting by the vote counts in **descending** order. Print `results` after sorting. [4]

Write program code to identify and output the artist(s) with the highest number of votes. Note that there can be artists sharing the same number of votes. [2]

Task 1.3

Write a function `task1_3` that uses binary search, modified to identify **all** artists that have received exactly `t` votes. Return a list containing these artists; otherwise return an empty list. Show the output for the cases where `t` is 16 and 2 respectively. [5]

Save your Jupyter Notebook for Task 1.

2 Name your Jupyter Notebook as:

TASK2_<your name>_<centre number>_<index number>.ipynb

Knowing how important data structures are in real life tech interviews in assessing problem solving skills, a Computing student decides to implement a hybrid data structure integrating binary search tree, linked list, stack and queue to manage players' game scores.

Implement a binary search tree (BST) of stack or queue. Each BST node comprises the following:

- `ds`: a variable having the value 'S' (stack) or 'Q' (queue)
- `score`: game score
- `data`: stores player ids with the same score
 - o if `ds` is 'S', `data` is a stack implemented using linked list
 - o if `ds` is 'Q', `data` is a wraparound queue implemented using array
- `left`: pointer to left subtree
- `right`: pointer to right subtree

You may assume that a player only has one game score.

Each stack or queue can store a maximum of 10 player ids with the same game score.

- a stack has a `top` pointer where insertion (`push`) and deletion (`pop`) happen, as well as a `size` attribute representing the number of items in the stack
- a queue has a `front` pointer where deletion (`dequeue`) happens and a `rear` pointer where insertion (`enqueue`) happens, as well as a `size` attribute representing the number of items in the queue

Each data structure class has the following methods:

- `BSTNode`: constructor/initialiser
- `BST`: constructor/initialiser, `insert()`, `search()`, `inorder()`
- `LinkedListNode`: constructor/initialiser
- `LinkedList`: constructor/initialiser, `insert()`
- `Stack`: constructor/initialiser, `push()`, `pop()`, `display()`
- `Queue`: constructor/initialiser, `enqueue`, `dequeue`, `display()`

Task 2.1

Write program code to create the classes `BSTNode`, `BST`, `LinkedListNode`, `LinkedList`, `Stack` and `Queue` with the associated attributes and methods. [16]

Task 2.2

Write program code to insert data from `scores.txt` into the `BST` using the `insert()` method. Verify that the data has been inserted successfully using the `inorder()` method. [4]

Task 2.3

Write program code to search for the scores 75 and 53. Output

- the player id(s) for a successful search
- "Not found" for an unsuccessful search [3]

Task 2.4

Write program code to implement and test a `maximum( )` method to determine the highest score and output the player id(s) with this score. [4]

Save your Jupyter Notebook for Task 2.

3 Name your Jupyter Notebook as:

TASK3_<your name>_<centre number>_<index number>.ipynb

QuickBuy is an online retailer that wants to improve its customer shopping experience. Customers can leave reviews and ratings for products they have purchased.

QuickBuy needs to keep track of its data using both a relational database (SQLite) and a NoSQL database (MongoDB) and object-oriented programming techniques.

The following shows the relational schema for the SQLite database quickbuy.db with three tables:

- customers (customer_id, customer_name, email)
 - products (product_id, product_name, category)
 - orders (customer_id#, product_id#, order_date, quantity)
- # denotes foreign key

Task 3.1

Write program code to create the three tables for the relational database. [6]

Task 3.2

Write program code to insert records from customers.txt, products.txt and orders.txt into the three tables. [5]

Task 3.3

Write program code to get and output the total quantity of products ordered by each customer. [3]

Task 3.4

Write program code to get and output the number of orders and total quantity for each product category for orders made in the year 2023. [4]

Task 3.5

Write program code to insert data from reviews.json to a quickbuy MongoDB database reviews collection. [3]

Task 3.6

Write program code to find and output all highly rated products (at least rating 4) with qualitative reviews. [3]

Task 3.7

Write program code to find and output all the reviews for product P5 and calculate and output its average rating correct to 1 decimal place. [4]

Save your Jupyter Notebook for Task 3.

- 4 A student intends to create a web application that allows users to upload image files. To add a layer of security, she wants to encrypt each image filename using one of the two cipher methods described below.

Task 4.1

This variant of the **autokey cipher** applies shifts to characters based on both a keyword and the previously encrypted characters.

Each letter in the keyword corresponds to a shift value: 'a' means no shift, 'b' means a shift of 1, up to 'z' which means a shift of 25 (note that the case of the letters in the keyword does not matter). The keyword is used to encrypt the first few characters of the main text. After the keyword is exhausted, each subsequent letter is encrypted using the previous encrypted letter. Non-alphabetical characters are not encrypted.

For example, if we were to encrypt the text “Hello everybody!” using the keyword “key”, we obtain the result “Rijui mhlcabpsq!”:

Text	H	e	l	l	o		e	v	e	r	y	b	o	d	y	!
Encrypt using	k	e	y	j	u		i	m	h	l	c	a	b	p	s	
Result	R	i	j	u	i		m	h	l	c	a	b	p	s	q	!

If the keyword “keyword” instead, the result would be “Rijhc vyc trsgj h!”.

Text	H	e	l	l	o		e	v	e	r	y	b	o	d	y	!
Encrypt using	k	e	y	w	o		r	d	y	c	t	r	s	g	j	
Result	R	i	j	h	c		v	y	c	t	r	s	g	j	h	!

Write a function `task4_1(main_text, keyword)` that:

- takes two non-empty strings, `main_text` and `keyword`
- applies the cipher described above to `main_text` and returns the resultant string.

Test your function by printing the output for the two examples above.

Save your program code as

`TASK4_1_<your name>_<centre number>_<index number>.py`

[7]

Task 4.2

A **substitution cipher** replaces each letter in the main text with another letter.

The key is a mapping of each letter of the alphabet to another letter, e.g. if the key is “qwertyuiopasdfghjklzxcvbnm”, then “a” in the main text is replaced by “q”, “b” is replaced by “w”, “c” is replaced by “e”, “d” is replaced by “r”, and so on.

Both upper and lower case letters should be handled, and non-alphabetical characters should be left unchanged.

Using the key “qwertyuiopasdfghjklzxcvbnm” (the letters in the 3 rows of your keyboard), the text “Life is hard. Why not make it harder?” becomes “Soyt ol iqkr. Vin fgz dqat oz iqkrtk?”.

Write a function `task4_2(main_text, key)` that:

- takes a string `main_text` and a 26-letter string `key`
- applies substitution cipher to `main_text` and returns the resultant string.

Test your function by printing the output for the example above, as well as the output of the text “I’ll be 100% ready to take the ‘A’ Levels!”, both using the key “qwertyuiopasdfghjklzxcvbnm”.

Add your program code to

`TASK4_2_<your name>_<centre number>_<index number>.py` [4]

Task 4.3

Write a function `task4_3` that returns a random 26-letter key for the substitution cipher, where all the letters in the key are distinct letters from ‘a’ to ‘z’. You should use basic loops and conditionals to achieve this. Do **not** use the `shuffle()` method from the `random` module.

Add your program code to

`TASK4_3_<your name>_<centre number>_<index number>.py` [3]

Task 4.4

The student eventually decides to use the substitution cipher to encrypt filenames.

Create a single-page web application that allows users to upload image files securely.

- Users should be able to upload image files to an `/uploads` folder in the server.
- Ensure that the filename is safe and free from potentially dangerous characters.
- Filename encryption:
 - Users can enter a 26-letter substitution cipher key in a text box.
 - If no key is provided (or length of key is incorrect), a random key would be created.
 - The filename (excluding the extension) is encrypted using a substitution cipher with the key.
- Upon successful form submission, the web page should indicate that the upload is complete, showing the encrypted filename (with its extension) along with the cipher key used.
- Store the timestamp (current system datetime) and the encrypted filename (with its extension) in a comma-separated text file `uploads.txt`.
- When the page loads, for the case where there are existing entries in the text file, display a table with 3 columns: the first column showing the timestamp, the second showing the encrypted filename (with file extension) and the third showing the corresponding image.

8

Test the web application by uploading the 2 image files provided (with extensions .jpg, .png). Use the cipher key “qwertyuiopasdfghjklzxcvbnm” for all uploads.

Save the project files (including “uploads.txt”) and sub-folders for this task in a folder named

TASK4_4_<your name>_<centre number>_<index number>

[10]