

HWA CHONG INSTITUTION
C2 PRELIMINARY EXAMINATION 2024

COMPUTING

Higher 2

19 AUG 2024

Paper 2 (9569 / 02)

1400 – 1700 hrs

Additional Materials:

Electronic version of `paint.csv` data file

Electronic version of `TASKS.txt` data file

Electronic version of `CUSTOMER.txt` data file

Electronic version of `FLIGHT.txt` data file

Electronic version of `TICKET.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**.

Instruction to candidates:

Your program code and output for each of Task 1 to 4.3 should be saved in a single .ipynb file using Jupyter Notebook. 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.

1 Name your Jupyter Notebook as

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

The task is to implement columnar transposition cipher to encode a message.

The columnar transposition cipher is a type of transposition cipher in which the plain text is written into a grid of rows and columns based on a keyword. The cipher text is formed by reading the characters off the grid column by column in the alphabetical order of the keyword's letters. This method rearranges the characters of the plaintext, making it difficult to decipher without knowing the specific key and the order of rearrangement.

The following steps describe how columnar transposition is used to encode a plain text to a cipher text.

- All spaces between each word in the plain text are replaced by 'z'.
- Write the plain text in rows of length equal to the number of letters in the keyword in a grid. If the plain text does not fit perfectly in the row of the grid, fill the empty spaces with 'x'.

For example, if the plaintext is "computing is an interesting subject", and the keyword is "ALGORITHM", the grid will look as follows:

A	L	G	O	R	I	T	H	M
c	o	m	p	u	t	i	n	g
z	i	s	z	a	n	z	i	n
t	e	r	e	s	t	i	n	g
z	s	u	b	j	e	c	t	x

3

- Assign a number to each letter of the keyword in alphabetical order. The cipher text is obtained by reading down the columns in ascending order of the numbers.

1	5	2	7	8	4	9	3	6
A	L	G	O	R	I	T	H	M
c	o	m	p	u	t	i	n	g
z	i	s	z	a	n	z	i	n
t	e	r	e	s	t	i	n	g
z	s	u	b	j	e	c	t	x

Cipher text: cztzmsruninttnteoiesgngxpzebuasjizic

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:

Task 1.1

Write a function `task1_1(key, message)` that takes in two arguments. The function uses columnar transposition to encode the message by using the key and returns a cipher message. [11]

Task 1.2

Write program code to encode the message "computing is an interesting subject" with key "ALGORITHM" and print the cipher message. [1]

Save your Jupyter Notebook for Task 1.

2 Name your Jupyter Notebook as

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

For this question, you must **not** use any built-in search or sort function.

When RGB values are given out of 255, each number represents the intensity of the respective colour channel (Red, Green, or Blue) on a scale from 0 to 255.

When combined, these three values (R, G, B) define a specific colour in the RGB colour space. For example, RGB (255, 0, 0) represents pure red, RGB (0, 255, 0) represents pure green, and RGB (0, 0, 255) represents pure blue. Different combinations of these values create the wide spectrum of colours.

A paint company has a catalogue of 15 colours and their RGB values are stored in a csv file `paint.csv`. For example, 3, 126, 251, ocean means that the colour named ocean is RGB (3, 126, 251).

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 2.1
Program Code
```

Output:

Task 2.1

Write a function `read_csv(filename)` which

- reads a csv file, `filename`
- stores the records in a 2D-array with 4 columns `red`, `green`, `blue` and `name`
- stores `red`, `green` and `blue` as integers
- stores `name` as a string
- returns the array.

Use the following code to test your function.

```
colours = read_csv("paint.csv")
print(len(colours))
print(colours)
```

[4]

Task 2.2

Write a function `find(array, r, g, b)` which

- linearly searches `array` for a colour with `r, g, b` values
- returns name or `'nil'` if the colour is not found.

Use the following code to test your function.

```
print(find(colours, 253, 253, 5))  
print(find(colours, 5, 253, 253))
```

 [4]**Task 2.3**

When a colour has a larger value for their Red colour channel, we say that the colour is more red. Similarly, colours can be more blue and more green.

Write a function `rgb(array, colour)` to implement a bubble sort algorithm which

- sorts `array` in descending order from the highest to lowest intensity by the `colour` provided
- takes `colour` as one of the values `'red', 'green'` or `'blue'`
- prints an appropriate error message and returns an empty array if an invalid `colour` is given
- returns the sorted array.

Use the following code to test your function.

```
print(rgb(colours, 'yellow'))  
print(rgb(colours, 'blue'))
```

 [6]

Task 2.4

As the colour intensity increases, the brightness also increases, we say that a colour is brighter when the total of their three colour channels gives a larger value.

Write a function `bright(array)` to implement a quick sort algorithm which

- sorts `array` in descending order from the brightest colour to the darkest colour
- returns the sorted array.

Use the following code to test your function.

```
print(bright(colours))
```

 [6]

Task 2.5

The paint company wants to create a menu which allows their clients to choose their preferred colour and recommends paint colours based on their preference.

Write a function `menu()` which

- reads the file `'paint.csv'`
- allows the user to choose their preference: red, green, blue, dark or bright
- repeats the prompt until a valid input is received
- prints out the name of the top 3 colours which matches the preference.

Use your function to show the recommended colours for **three** different users who prefer red, bright and dark colours respectively.

 [6]

Save your Jupyter Notebook for Task 2.

3 Name your Jupyter Notebook as

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

The task is to implement a task scheduler application using a linked list to store tasks in a priority queue to manage the execution order.

Tasks should execute in order of their priority, with higher priority tasks taking precedence. Additionally, tasks with the same priority should be executed in a first-come-first-serve manner.

The class `TaskNode` contains three attributes:

- `taskName`: Unique task name.
- `priority`: An integer priority of the task, with a higher number indicating higher priority.
- `next`: Reference to the next task node in the linked list.

The class `TaskNode` has the following method:

- `__str__()`: Returns a string representation of the task details (`taskName`, `priority`).

The class `PriorityQueue` contains one attribute:

- `head`: Reference to the first node in the queue.

The class `PriorityQueue` contains the following methods:

- `viewPendingTasks()`: Displays all tasks that are pending.
- `enqueue(taskNode)`: Adds a task node to the queue in the correct position based on its priority.
- `dequeue()`: Removes and returns the task node from the front of the queue. If the queue is empty, returns `None`.
- `isEmpty()`: Returns `True` if the queue is empty, or `False` otherwise.
- `delete(taskName)`: Deletes a task with the given `taskName`. Returns `True` if successful, or `False` otherwise.
- `search(taskName)`: Searches for a task node with the given `taskName` and returns it if found; else returns `None`.
- `updatePriority(taskName, newPriority)`: Updates the priority of a task with the given `taskName` and reorders the queue if necessary.

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 3.1
Program Code
```

Output:

Task 3.1

Write the program code for the `TaskNode` class. [2]

Task 3.2

Write the program code for the `PriorityQueue` class, including the following methods:

- `viewPendingTasks()`
- `enqueue(taskNode)`
- `dequeue()`
- `isEmpty()` [10]

A text file named `TASKS.txt` containing task data in the following format:
`taskName, priority`

In a new cell, write the program code to:

- create a `PriorityQueue` object
- enqueue all tasks read from `TASKS.txt`
- display all pending tasks
- dequeue **two** tasks and display the remaining pending tasks.

Test your program and show the output. All outputs should have appropriate messages to indicate what they are showing. [4]

Task 3.3

Amend your code for **Task 3.2** to include the following methods in the `PriorityQueue` class:

- `delete(taskName)`
- `search(taskName)`
- `updatePriority(taskName, newPriority)` [10]

In a new cell, write the program code to:

- search for 'Task 1' and print the result
- delete 'Task 5' and display the updated queue
- update the priority of 'Task 4' to 4 and display the updated pending tasks.

Test your program and show the output. All outputs should have appropriate messages to indicate what they are showing. [3]

Save your Jupyter Notebook for Task 3.

4 Name your Jupyter Notebook as

TASK4_<your name>_<centre number>_<index number>.ipynb

An airline company currently keeps paper records about their customers, the daily flights and the tickets sold. The manager wants to create a suitable database to store the data.

The database TRIP will have the following tables:

Flight:

- FlightNo – unique code for the flight, for example TR425
- DepartCity – the departure city of the flight
- ArrivalCity – the arrival city of the flight
- DepartTime – the departure time of the flight, for example 2220
- ArrivalTime – the arrival time of the flight, for example 2335

Customer:

- CustomerNo – unique integer number per customer, for example 12
- Name – the name of the customer
- Gender – the gender of the customer
- DOB – the date of birth of the customer, for example 21041997 means 21 Apr 1997

Ticket:

- TicketNo – unique integer number per ticket, for example 1010
- CustomerNo – the customer's unique number
- FlightNo – the unique flight code of the ticket
- Date – the date of flight on the ticket
- Seat – the seat category of the ticket, for example Economy

For each of the sub-tasks, 4.1 to 4.3, 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 4.1
Program Code
```

Output:

Task 4.1

Write a Python program that uses SQL code to create the database TRIP with the three tables given. Define the primary and foreign keys for each table. [4]

Task 4.2

The text files `FLIGHT.txt`, `CUSTOMER.txt` and `TICKET.txt` store the comma-separated values for each of the tables in the database. Write a Python program to read in the data from each file and store the data in the correct place in the database. [3]

Task 4.3

The company has rescheduled flight TR808's departure time to 18:15 and arrival time to 00:15. Write a Python program to update the database with this information, then display all the flights departing from Singapore in ascending order of departure time.

A customer named Shao Xingjuan wants to cancel her tickets numbered 1023 and 1024. Write a Python program to update the database with this cancellation. [7]

Save your Jupyter Notebook for Task 4.

Task 4.4

The company manager wants to filter the ticket data by Date and display the results in a web browser. Write a Python program and the necessary files to create a web application that:

- Receives a Date string from a HTML form,
- Returns an HTML document that enables the web browser to display a table tabulating the customer's name, departure city, arrival city and the seat category for tickets sold on that date.

Save your Python program as:

`TASK4_4_<your name>_<centre number>_<index number>.py`

with any additional files/subfolders in a folder named:

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

Run the web application with the date entered as '08082024'.

Save the web page output as:

`TASK4_4_<your name>_<centre number>_<index number>.html` [13]

BLANK PAGE