RIVER VALLEY HIGH SCHOOL
General Certificate of Education Advanced Level
Higher 2
Common Test 3

COMPUTING

9569/02

Paper 2 (Lab-based)

7 July 2023

3 hours

Additional Materials:

Electronic version of the following data file

- `Task1_template.ipynb`
- `Task2a_template.ipynb`
- `Task2b.txt`
- `A folder "data_files" consists of:`
  - `characters.csv`
  - `equipment.csv`
  - `relic_sets.csv`
  - `relics.csv`
  - `helper_for_Task3_and_4.txt`

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 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 of 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 **15** printed pages and **1** blank page.

1 Euclidean distance matrix

Task 1.1

Write the function `generate_points` which takes a positive integer `n` as input and return `n` randomly generated cartesian points `(x, y)` in a list of lists. The function must ensure that the number of points randomly generated are **evenly** distributed in the following 4 quadrants. You can assume that the positive integer `n` is always divisible by 4.

A. x ranges value from 1 – 50 and y ranges from 1 – 50
B. x ranges value from 51 to 100 and y ranges from 1 – 50
C. x ranges value from 1 – 50 and y ranges from 51 to 100
D. x ranges value from 51 to 100 and y ranges from 51 to 100

For example,

```
>>> generate_points(8)
>>> [[70, 23], [22, 58], [27, 71], [100, 74], [26, 6], [58,
45], [66, 95], [26, 13]]
```

Take note that:

- [26, 6] and [26, 13] are in quadrant A
- [70, 23] and [58, 45] are in quadrant B
- [22, 58] and [27, 71] are in quadrant C
- [100, 74] and [66, 95] are in quadrant D

[3]

Task 1.2

Write the function `test_1_1` which takes a list of points of the same output format in **Task 1.1** as input and returns `True` if the points are evenly distributed in the 4 quadrants and `False` otherwise.

For example:

```
>>> test_1_1([[16, 85], [88, 41], [50, 36], [80, 70]])
>>> True
>>> test_1_1([[16, 85], [88, 41], [50, 36], [80, 20]])
>>> False
```

[3]

Task 1.3

Write the function `euclidean_distance` which takes 2 points and returns the integer value of the Euclidean distance between the 2 points (discards all decimal points). The Euclidean distance, `d`, is calculated using the formula below.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

For example:

```
>>> euclidean_distance([1, 2], [4, 6])
>>> 5
```

[2]

Task 1.4

Write the function `distance_matrix` which takes a list of points of the same output format in **Task 1.1** as input and returns a Euclidean distance matrix of these points.

A Euclidean distance matrix is a `n` x `n` matrix that contains the distances between all pairs of points in a given set. The distance between two points is calculated using the Euclidean distance formula. For this Euclidean distance matrix, you only need to store the integer value of the Euclidean distances.

For example:

```
>>> distance_matrix([[52, 67], [89, 20], [38, 79], [9, 13]])
>>> [[0, 59, 18, 69], [59, 0, 77, 80], [18, 77, 0, 72], [69, 8
0, 72, 0]]
```

Take note that the size of the input list of points can vary.

The table below explains Euclidean distance matrix using the example above.

Points	**[52, 67]**	**[89, 20]**	**[38, 79]**	**[9, 13]**
[52, 67]	0	**59**	18	69
[89, 20]	**59**	0	77	80
[38, 79]	18	77	0	72
[9, 13]	69	80	72	0

In the distance matrix above, the Euclidean distances from [52, 67] to [89, 20] and from [89, 20] to [52, 67] are the same, both have a value of 59. This explains why the distance matrix is a reflection along the diagonal line. Moreover, the distance between the same point is 0. This explains the 0s in the distance matrix.

[4]

Task 1.5

Write the function `calculate_total_distance` which takes a tuple of point indices `path` and a distance matrix `d_matrix` and as inputs and returns the total distance needed to travel through the path.

Using the same distance matrix in the previous example.

```
>>> d_matrix = [[0, 59, 18, 69], [59, 0, 77, 80], [18, 77, 0,
72], [69, 80, 72, 0]]
>>> path = (1, 0, 2, 3)
>>> calculate_cost(path, d_matrix)
229
```

The following explains how `229` is calculated.

A path is made up of a tuple of indices which determines how the calculation of the total distances is done. The path makes a loop which means that it includes the sub-path from the ending point to the starting point.

For example,
`path = (1, 0, 2, 3)`
The above means that `path` is made up of following sub-paths in the same order:

- from point 1 to point 0
- from point 0 to point 2
- from point 2 to point 3
- from point 3 to point 1 (go back to starting point)

Using the distance matrix `d_matrix` illustrated below by replacing the actual x and y values of a point with indices 0, 1, 2 and 3.

Points	**0**	**1**	**2**	**3**
0	0	59	**18**	69
1	**59**	0	77	80
2	18	77	0	**72**
3	69	**80**	72	0

The distance from point 1 to point 0 is 59. The distance from point 0 to point 2 is 18. The distance from point 2 to point 3 is 72. The distance from point 3 to point 1 is 80. Hence, the total distance is 59+18+72+80 = 229.

[4]

Task 1.6

Write the function `insertionSort` which takes a list of paths, `paths`, and a distance matrix, `d_matrix`, as inputs. It sorts the paths in `paths` in ascending order based on the total distance covered in each path in `paths`. You must use insertion sort to perform this function.

For example,

```
>>> paths = [(1, 2, 0, 3), (0, 2, 1, 3), (1, 0, 2, 3), (3, 2,
1, 0)]
>>> insertionSort(paths, d_matrix)
>>> paths
[(1, 0, 2, 3), (1, 2, 0, 3), (0, 2, 1, 3), (3, 2, 1, 0)]
```

In the example above, the total distance covered for each path is as follows:

```
Total distance      Path
229                 (1, 0, 2, 3)
244                 (1, 2, 0, 3)
244                 (0, 2, 1, 3)
277                 (3, 2, 1, 0)
```

Take note that the size of each path and `d_matrix` can vary.

[6]

2a Customized Linked List

Task 2.1

The class `Node` is implemented for you. Complete the class `Linkedlist` based on the description below.

[10]

Class function	Description

`Linkedlist(compare_func)`	This is the constructor for the class `Linkedlist`. It takes a function object `compare_func` as input.
	
	`compare_func` is a function object that may be called repeatedly during the `insert(data)` operation. It helps to determine the position in which the new data is to be inserted by comparing the new data value with a current data value in the linked list and returns a Boolean value.
	
	For example, if the function below is used to initialize the `Linkedlist` object, the linked list will behave like a sorted linked list.
	
	`def dataIsSmaller(curr_data, new_data):`
	`  return curr_data > new_data`
	
	`>>> list1 = Linkedlist(dataIsSmaller)`
	`>>> list1.insert(4)`
	`>>> list1.insert(1)`
	`>>> list1.insert(3)`
	`>>> list1.insert(2)`
	`>>> list1.returnAsList()`
	`[1,2,3,4]`
	
	By varying the `compare_func`, you can decide how the new data is inserted into the linked list in the following possible ways.
	• insert new data at the head of the linked list
	• insert new data at the tail of the linked list
	• insert new data in ascending order
	• insert new data in descending order
	
	The constructor of class `Linkedlist` is already implemented for you.

`insert(data)`	This **recursive** function inserts the new data, `data`, into the linked list as a node. *Note: `compare_func` must be used to determine how the new data is inserted. If you do not know how to use it, implement a normal sorted linked list that will pass `test1()`, but marks will be deducted.*

`delete(data)`	This **iterative** function deletes the first node in the linked list with the value equal to `data`. If the node does not exist, the function does nothing.
`display()`	This function prints out the content of the linked list. This is implemented for you.
`returnAsList()`	This function returns the content of the linked list as a list. This is implemented for you.

Save your program as *Task2a.ipynb*.

Task 2.2

Define an appropriate `compare_function` such that the linked list will always insert the new item at the head of the linked list. [1]

Task 2.3

Define an appropriate `compare_function` such that the linked list will always insert the new item at the tail of the linked list. [1]

2b Choose Your Own Ending Adventure

*Take note that **Task 2b** is an independent question. It has no relation to **Task 2a**.*

Your task is to implement the “Choose your own ending adventure” game using socket programming. In this game, the server program sends the adventure scenario and 2 possible options to the client program. The server program then sends the next adventure scenario based on the option chosen by the client program.

The scenario used by the server program can be found in “*Task2b.txt”*. Each line in the file consists of an adventure scenario and two options in the following format.

`<line>-<scenario>,<line>-<1st option>,<line>-<2st option>`

For example, the first three lines of *“Task2b.txt”* are as follows.

1-You arrive at the jungle and start walking. You come across a fork in the road.,2-Take the left path,3-Take the right path
2-You take the left path and come across a river.,4-Try to swim across,5-Look for a bridge
3-You take the right path and come across a cave.,6-Enter the cave,9-Keep walking
…

The server program begins the game using the first line. It sends the following scenario and options to the client.

Scenario
“*You arrive at the jungle and start walking. You come across a fork in the road.*”
Options
“Take the left path”
“Take the right path”

If the client chooses *“Take the left path”*, the server program will go to line 2 and send the following scenario and options to the client.

Scenario
“You take the left path and come across a river.”
Options
“Try to swim across”
“Look for a bridge”

If the client chooses *“Take the right path”*, the server program will go to line 3 and send the following scenario and options to the client.

Scenario
“You take the right path and come across a cave.”
Options
“Enter the cave”
“Keep walking”

The server program repeats the above until the client chooses the option – “Give up”. Then both server and client program quit successfully.

Save your server and client program as *Task2b_server.ipynb* and *Task2b_client.ipynb*

A screenshot of the client program of 1 possible outcome is below.

```
You arrive at the jungle and start walking. You come across a fork in the road.
1) Take the left path
2) Take the right path
Choose 1 or 2: 2
You take the right path and come across a cave.
1) Enter the cave
2) Keep walking
Choose 1 or 2: 2
Your adventure ends with no excitment.
1) Start Over
2) Give Up
Choose 1 or 2: 2
Quitted!
```

[20]

3 Star Road OOP

Hint: The following text file contains strings and text which you can simply copy-paste when attempting Task 3 and 4: `helper_for_Task3_and_4.txt`.

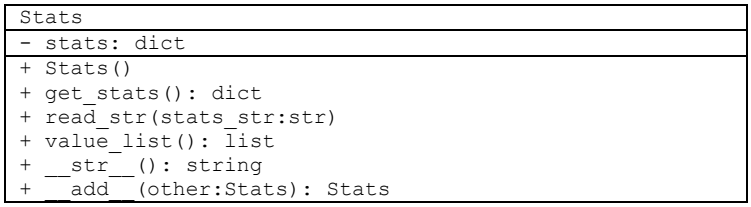
Star Road is a game created by the company Mohoyo. It is a turn-based game which allows you to control characters and battle with enemies, in Task 3 and 4, you are to aid the company to create an OOP class and a simple web application related to this game.

First, you need to create a class named Stats. The class supposed to store related stats for a character or a relic item, which would enhance the characters' statistics. The class contains 1 private attribute `stats` of type `dictionary`.

A typical input and output string would look like the following format (as one line):

```
'HP:1000;ATK:100;DEF:100;Speed:100;CRIT Rate:5;CRIT DMG:50;Break Effect:0;Effect
RES:0;Outgoing Healing:0;Energy Regen Rate:0;HP%:0;ATK%:0;DEF%:0'
```

Below is an UML class diagram for your reference.



Implement the classes based on the following descriptions:

Attributes/Methods	Description
`Character Class`	
`- stats: dict`	A dictionary which stores the value of stats, which has an initial value as `None`. Such as: `{'HP': None, 'ATK': None, 'DEF': None, ...}`
`+ Stats()`	Constructor of `Stats` class, which initializes the dictionary with all stats with `None` values.
`+ get_stats(): dict`	Getter method, which returns the current dict.
`+ read_str(stats_str:str)`	Method which read the stat from a `stats_str`, and store them into the dictionary based on the stat names and stat values. a) If the stat name is not currently found in the list of stats provided, do not add it (do not create any additional key-value pair in the dictionary). b) If the stat name is found, but the stat value is empty string `''` or `None`, treat it as a value of `0`. c) If the stat name is found, and stat value is an `integer` value, update the corresponding key-value pair in dictionary. If stat value is not an `integer` value, do nothing.

	For example, if the `stats_str` is `"HP:1000;ATK:None;DEF:;Speed:one"` Then it should update the key values pairs of `HP`, `ATK` and `DEF` to `1000`, `0` and `0` respectively and ignore `Speed`.
`+ value_list(): list`	Return a list of stat values based on the stat names stored in the `stat_name_list`.
`+ __str__(): string`	String method for the class, to return a string with stat names and values in the following format '`name:value`', separate each stat by using an `';'`: `HP:1000;ATK:0;DEF:None;…`
`+ __add__(other:Stats): Stats`	Add all the stat values in the current `Stats` object (self) with another `Stats` object. Then return a `Stats` object as the result. If the stat value of one of the object is `None`, treat it as `0` when adding with the other object's stat value. If the stat values of both of the object are `None`, let the value of the stat returned be `None`.

Save your program code as
`Task3.py` [12]

4 Star Road Web app

In the Star Road game, there are relic sets, which are equipment items which characters can equip themselves with. Each relic sets have their unique set effects, which will boost the power of the characters wearing them.

For each relic sets, relics can be generated with random stats. Even if two relics are with the same name, they might have different stats associated with. For example, one of them may have stats that boost the attack value, the other relic may have stats that boost the energy regeneration rate.

Characters can choose to wear up to 6 relics, one at each position such as head, body, feet, etc.

You are tasked to design a relational database model and a web app to store and display the related information. You may assume that all inputs given are valid.

Task 4.1

The following information of each `Character` is stored:
`ID` – auto increment integer value to keep track of ID of the Character.
`name` – name of the Character.
`element` – element of the Character's attacks, it should be one of the values:
`'Physical', 'Fire', 'Ice', 'Lightning', 'Wind', 'Quantum', 'Imaginary'`
`path` – path of the Character, it should be one of the values:
`'Destruction', 'The Hunt', 'Erudition', 'Harmony', 'Nihility', 'Preservation', 'Abundance'.`
`Stats` – stats of a Character, which is a long string of text.

The following information of each `RelicSet` is stored:
`ID` – auto increment integer value to keep track of ID of the relic set.
`Name` – name of the relic set.
`SetEffect` – set effects when wearing more than 1 pieces of the same set.

The following information of each `Relic` is stored:
`ID` – auto increment integer value to keep track of the Relic.
`Name` – name of the relic.
`SetID` – Relic Set ID.
`Position` – position to be worn by the Character, it should be one of the values:
`'Head', 'Body', 'Hand', 'Feet', 'Planar Sphere', 'Link Rope'`
`Stats` – stats of a Relic, which is a long string of text.

The following information of each `Equipment` is stored:
`CharacterID` – Character ID.
`RelicID` – Relic ID.

The information is to be stored in above mentioned tables:
`Character`
`RelicSet`
`Relic`
`Equipment`

Task 4.1

Create an SQL file called `Task4_1.sql` to show the SQL code needed to create the database `star_road.db` with the above tables.

The tables `Character, RelicSet, Relic` must use `ID` as its primary key. The table `Equipment` must use `CharacterID` and `RelicID` as its primary key.

The `SetID` in table `Relic` must refer to `ID` in `RelicSet` table as foreign key. `CharacterID` and `RelicID` in table `Equipment` must refer to `ID` in `Character` table and `ID` in `Relic` table as foreign keys.

Save your SQL code as
`Task4_1.sql`

[6]

Task 4.2

The following files contains the past data 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.

`characters.csv`
`relic_sets.csv`
`relics.csv`
`equipments.csv`

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

Save your program code as
`Task4_2.py` [5]

Task 4.3

You are tasked to implement a function to display the number of relics of each relic set that are currently equipped by the characters. Query and display a list of data with the following fields as shown in the table, sorted in the descending order by the Count value.

Note: *To make it simple, if 1 character wears 4 relic pieces from the same relic set, count 4 times.*

Set Name	Count
…	…

Write the SQL code required.

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

Task 4.4

You are required to implement a function to search and display a summary of the equipped relics of a character based on the character name.

You should display the character name, followed by a table containing the following columns:
`Relic Name, Position, Relic Set Name, Set Effect, Stats`

Here's a sample result page for your reference:

Result Page

The character is Seele.

Relic Name	Position	Relic Set Name	Set Effect	Stats
Genius's Ultraremote Sensing Visor	Head	HP:112;ATK:None;DEF:16;Speed:2;CRIT Rate:None;CRIT DMG:None;Break Effect:5;Effect RES:None;Outgoing Healing:None;Energy Regen Rate:None;HP%:None;ATK%:3;DEF%:None	Genius of Brilliant Stars	Two Set Effect: Increases Quantum DMG by 10%. | Four Set Effect: When the wearer deals DMG to the target enemy, ignores 10% DEF. If the target enemy has Quantum Weakness, the wearer additionally ignores 10% DEF.
Genius's Frequency Catcher	Body	HP:33;ATK:56;DEF:None;Speed:2;CRIT Rate:2;CRIT DMG:None;Break Effect:None;Effect RES:None;Outgoing Healing:None;Energy Regen Rate:None;HP%:None;ATK%:None;DEF%:4	Genius of Brilliant Stars	Two Set Effect: Increases Quantum DMG by 10%. | Four Set Effect: When the wearer deals DMG to the target enemy, ignores 10% DEF. If the target enemy has Quantum Weakness, the wearer additionally ignores 10% DEF.
Genius's Metafield Suit	Hand	HP:33;ATK:None;DEF:None;Speed:None;CRIT Rate:None;CRIT DMG:5;Break Effect:None;Effect RES:None;Outgoing Healing:None;Energy Regen Rate:None;HP%:3;ATK%:None;DEF%:4	Genius of Brilliant Stars	Two Set Effect: Increases Quantum DMG by 10%. | Four Set Effect: When the wearer deals DMG to the target enemy, ignores 10% DEF. If the target enemy has Quantum Weakness, the wearer additionally ignores 10% DEF.
Genius's Gravity Walker	Feet	HP:33;ATK:16;DEF:None;Speed:None;CRIT Rate:None;CRIT DMG:None;Break Effect:None;Effect RES:None;Outgoing Healing:None;Energy Regen Rate:None;HP%:3;ATK%:None;DEF%:4	Genius of Brilliant Stars	Two Set Effect: Increases Quantum DMG by 10%. | Four Set Effect: When the wearer deals DMG to the target enemy, ignores 10% DEF. If the target enemy has Quantum Weakness, the wearer additionally ignores 10% DEF.
Herta's Space Station	Planar Sphere	HP:None;ATK:None;DEF:0;Speed:2;CRIT Rate:None;CRIT DMG:5;Break Effect:5;Effect RES:None;Outgoing Healing:None;Energy Regen Rate:None;HP%:None;ATK%:3;DEF%:None	Space Sealing Station	Two Set Effect: Increases the wearer's ATK by 12%. When the wearer's SPD reaches 120 or higher, the wearer's ATK increases by an extra 12%.
Herta's Wandering Trek	Link Rope	HP:None;ATK:None;DEF:None;Speed:None;CRIT Rate:None;CRIT DMG:5;Break Effect:None;Effect RES:3;Outgoing Healing:None;Energy Regen Rate:None;HP%:3;ATK%:None;DEF%:4	Space Sealing Station	Two Set Effect: Increases the wearer's ATK by 12%. When the wearer's SPD reaches 120 or higher, the wearer's ATK increases by an extra 12%.

Search Character Again

Save all files and folders under the directory `Task4_4`.

Run the web application with a character name input of `Seele`.
Then save the output of the program as `Task4_4.html`. [12]

Task 4.5 [Bonus – Challenge by Choice]

You are encouraged to copy and paste your files from `Task4_4` to the new directory `Task4_5` before attempting this task.

You are tasked to display the information of this page with better clarity and format:

- Separate the repeated set name and their set effects.
- Display stats in individual columns.
- Calculate the overall stats by adding the base stats of the character with all the stats the character has gained from the equipped relics.

Here's a sample page for character `Seele`:

Result Page

The character is Seele.

Set Name: Genius of Brilliant Stars

Set Effect: Two Set Effect: Increases Quantum DMG by 10%. | Four Set Effect: When the wearer deals DMG to the target enemy, ignores 10% DEF. If the target enemy has Quantum Weakness, the wearer additionally ignores 10% DEF.

Set Name: Space Sealing Station

Set Effect: Two Set Effect: Increases the wearer's ATK by 12%. When the wearer's SPD reaches 120 or higher, the wearer's ATK increases by an extra 12%.

Relic Name	Position	HP	ATK	DEF	Speed	CRIT Rate	CRIT DMG	Break Effect	Effect RES	Outgoing Healing	Energy Regen Rate	HP%	ATK%	DEF%
Seele	Character	126	87	49	115	5	50							
Genius's Ultraremote Sensing Visor	Head	112		16	2			5					3	
Genius's Frequency Catcher	Body	33	56		2	2								4
Genius's Metafield Suit	Hand	33					5					3		4
Genius's Gravity Walker	Feet	33	16									3		4
Herta's Space Station	Planar Sphere				2		5	5					3	
Herta's Wandering Trek	Link Rope						5		3			3		4
Overall		337	159	65	121	7	65	10	3	0	0	9	6	16

Search Character Again

Save all files and folders under the directory `Task4_5`.

[2]

- End of Paper -