

Name:		Index Number:		Class:	
-------	--	---------------	--	--------	--



DUNMAN HIGH SCHOOL

Preliminary Examination

Year 6

COMPUTING (Higher 2)

9569/02

Paper 2 (Lab-based)

25 August 2025

3 hours

Additional Materials: Insert

Electronic version of `IT_Devices.json` file

Electronic version of `read_json_sample.py` file

Electronic version of `datetime_sample.py` file

Electronic version of `GuessingPairGame.py` file

Electronic version of `SERVER.py` file

Electronic version of `CLIENT.py` file

Electronic version of `container_data.csv` file

Electronic version of `napfa_results.csv` file

READ THESE INSTRUCTIONS FIRST

Write your name, index number and class on the work you hand in.

Write in dark blue or black pen on both sides of the paper.

You may use an HB pencil for any diagrams or graphs.

Do not use staples, paper clips, glue or correction fluid.

Answer **all** the 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.

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.

Instructions to candidates:

Your program code and output for Task 1 and 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 Dunman High School wants to store records of its IT Devices in a NoSQL database. The list of devices (laptops & tablets) are recorded in the file `IT_Devices.json`.

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 [ ]: #Task 1.1
        Program code
```

Output:

Task 1.1

Write program code to insert the data from the file `IT_Devices.json` into a NoSQL database `DHS` under the collection `IT_Devices`. The program should clear the `IT_Devices` collection if it exists.

(Note: You may refer to the `read_json_sample.py` file for an example of how to extract data from JSON files.)

Retrieve and display all the documents in the collection `IT_Devices` after the insertion of data. [4]

Task 1.2

The school has decided to change all occurrences of the brand name `HP` to its full name, `Hewlett-Packard`. Write a program to replace all values of `HP` in the collection `IT_Devices` with `Hewlett-Packard`. [3]

Subsequently, extend the above program to retrieve and display all laptops that weigh less than 2 kg together with all tablets that have at least 8000 mAh Battery Capacity. Display all relevant devices in the following format: [4]

```
Serial No: LAP-001, Type: Laptop, Brand: Hewlett-Packard, Weight: 1.35
Serial No: TAB-002, Type: Tablet, Brand: Apple, Battery Capacity: 9720
```

Task 1.3

The school decided to move all of its data into a relational database (DHS.db). The database has three tables: one table to store the devices' common information, and separate tables for laptops and tablets to store their specific information.

Device (Serial_Number, Type, Brand, Cost, Date_Of_Purchase)

Laptop (Serial_Number, Weight, Screen_Size)

Tablet (Serial_Number, Battery_Capacity)

Device:

- Serial_Number – the unique device identifier, for example, LAP-001
- Type – the category of the device, either Laptop or Tablet
- Brand – the brand of the device, for example, Dell
- Cost – the cost of the device, for example, 2199
- Date_Of_Purchase – the purchase date of the device in YYYY-MM-DD

Laptop:

- Serial_Number – the unique device identifier, for example, LAP-001
- Weight – the weight of the laptop in kg, for example, 2.6
- Screen_Size – the screen size of the laptop in inches, for example, 17

Tablet:

- Serial_Number – the unique device identifier, for example, TAB-001
- Battery_Capacity – the battery capacity of the tablet in mAh, for example, 7500

Write a Python program that uses SQL code to create the database DHS.db with the three tables given. Define the primary and foreign keys for each table. [6]

Additionally, extend the above program to retrieve all data from the DHS NoSQL database's collection, IT_Devices, and store the data in the correct place in the relational database, DHS.db. [5]

Task 1.4

The school would like to keep a log of all the school devices that are older than 1000 days old.

Write program code to

- extract relevant data from the relational database, DHS.db
- calculate the age of each device (in days) by subtracting the purchase date from today's date
- write the Serial_Number, Type, Date_Of_Purchase and age of device (in days) of devices older than 1000 days old to a csv file with the name Old_devices.csv

The datetime library is built into Python and can be used to manipulate dates. See the example code shown in datetime_sample.py and use the datetime library to calculate the age of device (in days). [6]

Run your program.

Save your Jupyter Notebook for Task 1.

- 2** You created a new online game called the Guessing Pair Game where a human player (client program) plays against the computer player (server program). At the beginning of the game, a board of 2 rows & 4 columns is generated and the 8 cells are populated randomly with 4 pairs of numbers range between 1 and 9 (inclusive). The aim of the game is to find the same number on the board by choosing 2 cells in each turn. When a pair of numbers is found, points are added to the human/computer player based on the number revealed. The human player will always start first followed by the Computer. The game ends when all pairs have been found.

Study the sample program output in the next three pages to determine your code design, output format and socket protocol. User inputs are underlined and comments are shown after #. [Note: there is no need to display the comments in your programs]

Sample server program (Computer player):

```

-----
GUESSING PAIR GAME
-----

Waiting for player to join
-----
Revealed Board: [[3, 2, 5, 5], [3, 7, 7, 2]]
#Revealed Board is only shown in Server Program

Board:
X | X | X | X
X | X | X | X

Player chose: row 1: 0, col 1: 1, row 2: 1, col 2: 3
Player Score: 2
Computer Score: 0

Board:
X | 2 | X | X
X | X | X | 2

Computer chose: row 1: 1, col 1: 1, row 2: 1, col 2: 2
Player Score: 2
Computer Score: 7

Board:
X | 2 | X | X
X | 7 | 7 | 2

Player chose: row 1: 0, col 1: 2, row 2: 0, col 2: 3
Player Score: 7
Computer Score: 7

Board:
X | 2 | 5 | 5
X | 7 | 7 | 2

Computer chose: row 1: 1, col 1: 0, row 2: 0, col 2: 0
Player Score: 7
Computer Score: 10

```

```
Board:
 3 | 2 | 5 | 5
 3 | 7 | 7 | 2
```

Player lose!

```
-----
GAME ENDED
-----
```

Sample client program (Human player):

```
-----
GUESSING PAIR GAME
-----
Board:
 X | X | X | X
 X | X | X | X

INPUT first coordinates row number: 0
INPUT first coordinates column number: 1

INPUT second coordinates row number: 1
INPUT second coordinates column number: 3

Its a match! You scored 2 points
Player Score: 2
Computer Score: 0

Board:
 X | 2 | X | X
 X | X | X | 2

Computer chose: row 1: 1, col 1: 1, row 2: 1, col 2: 2
Its a match! Computer scored 7 points
Player Score: 2
Computer Score: 7

Board:
 X | 2 | X | X
 X | 7 | 7 | 2

INPUT first coordinates row number: 0 #cell already revealed so
INPUT first coordinates column number: 1 #prompt for another cell
INPUT first coordinates row number: 0
INPUT first coordinates column number: 2

INPUT second coordinates row number: 5 #user input is
INPUT second coordinates column number: 7 #out of range
INPUT second coordinates row number: 0 #second input is the same
INPUT second coordinates column number: 2 #as the first input
INPUT second coordinates row number: 0
INPUT second coordinates column number: 3

Its a match! You scored 5 points
Player Score: 7
Computer Score: 7
```

```

Board:
X | 2 | 5 | 5
X | 7 | 7 | 2

Computer chose: row 1: 1, col 1: 0, row 2: 0, col 2: 0
Its a match! Computer scored 3 points
Player Score: 7
Computer Score: 10

Board:
3 | 2 | 5 | 5
3 | 7 | 7 | 2

You lose!
-----
GAME ENDED
-----

```

Task 2.1

The Guessing Pair Game is implemented using Object-Oriented Programming.

The class `GuessingPairGame` contains four properties:

- `board` stores a 2D array (2 rows of 4 columns) with each cell containing a number
- `revealed` stores a 2D array (2 rows of 4 columns) tracking the visibility status of each corresponding cell in `board` (`True` indicates revealed, `False` indicates hidden)
- `player_score` stores the human player's score
- `computer_score` stores the computer's score

The class `GuessingPairGame` contains the following methods:

- `create_board()` generates the initial board which is a 2D array (2 rows of 4 columns) with cell values set to `-1`
- `create_revealed_board()` generates the initial revealed board which is a 2D array (2 rows of 4 columns) with cell values set to `False`
- `populate_board()` fills the board with 4 pairs of random numbers that range between 1 and 9 (inclusive). The placement of the numbers is also randomised
- `set_board(values)` fills the board according to the provided values string (example values string: `7,9,5,7,9,1,5,1`)
- `print_board()` displays the board. Shows `x` for cells that remain hidden and shows the actual value for revealed cells
- `board_in_string()` converts the board values into a string format, e.g. `7,9,5,7,9,1,5,1`
- `valid_choice(row, column)` returns `True` if the selected row and column are within bounds and the chosen cell is not yet revealed, otherwise returns `False`
- `play_move(row1, column1, row2, column2)` determines if the two selected cells contain the same number, and if they match, update their revealed status in the `revealed` board to `True` and returns the score equivalent to the number revealed. Returns `0` if the two cells do not match
- `is_game_over()` checks if all cells have been revealed. If all are revealed, returns `True`, otherwise returns `False`

You are provided with the template program for Guessing Pair Game, `GuessingPairGame.py`. Some of the codes have been provided for you. Complete the code for `populate_board()`, `play_move(row1, column1, row2, column2)` and `is_game_over()`.

[Note: DO NOT rename the file `GuessingPairGame.py`. Write your name as a comment on the first line of code in `GuessingPairGame.py`. [9]

Task 2.2

Using socket programming, complete the client program to:

1. establish connection to the server program
2. receive the board values in string format from the server program
3. load the received board values into its own `GuessingPairGame` instance using the relevant method
4. collect user input for the 2 coordinates. Request user to re-enter coordinates if the provided coordinates are invalid. Validate using the relevant method and conditions.
5. modify the `revealed board` and `player_score` using the relevant method or otherwise according to user input and show an appropriate message.
6. send the user selected coordinates in the appropriate format to the server program to modify its `revealed board` and `player_score`
7. verify if the game has ended using the relevant method
8. receive the 2 coordinates from the server program (Computer player's move)
9. update the `revealed board` and `computer_score` using the relevant method or otherwise according to computer player's selected coordinates and show an appropriate message.
10. verify if the game has ended using the relevant method
11. repeat Steps 4 to 10 until the game has ended
12. display whether Player wins or loses according to the score

You are provided with the completed server program and client template program, `SERVER.py` and `CLIENT.py` respectively. Complete the client program and rename as `CLIENT_<your name>_<centre number>_<index number>.py` [14]

Save all your files for Task 2.

3 Name your Jupyter Notebook as:

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

A container yard is the main area in a shipping port where shipping containers are stored. This is a large, organised space where containers are stacked and temporarily stored.

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 [ ]: #Task 3.1
        Program code
```

Output:

Task 3.1

A shipping port is planning to create a program to manage `ContainerNode` via `LinkedList` Data Structure. Each `ContainerNode` represents a shipping container.

The class `ContainerNode` contains four **private** attributes:

- `container_number` is the ID of the shipping container, for example, MSKU1234567
- `category_of_goods` is the type of goods stored, for example, Electronics
- `cargo_weight` is the weight of the goods stored, for example, 15000 kg
- `next` is the pointer that points to the next `ContainerNode`, initialised to `None`

The class `LinkedList` contains one **private** attribute:

- `head` is the pointer that points to the first `ContainerNode`, initialised to `None`

The class `LinkedList` has the following methods:

- `insert_to_end()` takes the `container_number`, `category_of_goods` and `cargo_weight` as parameters, creates a new `ContainerNode` instance and inserts it at the end of the `LinkedList`.
- `insert_to_front()` takes the `container_number`, `category_of_goods` and `cargo_weight` as parameters, creates a new `ContainerNode` instance and inserts it at the front of the `LinkedList`.
- `delete_from_end()` removes the last `ContainerNode` instance from the `LinkedList` and returns the deleted `ContainerNode` instance (to return string "Cannot delete - Empty" if `LinkedList` is empty)
- `delete_from_front()` removes the first `ContainerNode` instance in the `LinkedList` and returns the deleted `ContainerNode` instance (to return string "Cannot delete - Empty" if `LinkedList` is empty)

Write program code to declare the class `ContainerNode` and `LinkedList`. Ensure that each class contains a complete set of methods, including all the accessors and mutators. [13]

Task 3.2

The shipping port decided to manage the `ContainerNode` via `Stack` Data Structure instead.

Write a Python subclass `Stack` using `LinkedList` as its superclass. The subclass `Stack` has the following additional methods:

- `is_empty()` returns `True` if the `Stack` is empty, otherwise `False`
- `peek()` returns the `container_number` of the topmost `ContainerNode` instance. Return "No containers" if `Stack` is empty
- `display()` shows the `container_number` of all `ContainerNode` instances in the `Stack` using the below format (Note: the most recently added `ContainerNode` appears at the top). Return "No containers" if `Stack` is empty

```
OOLU2468135
CMAU7654321
MSKU1234567
```

[4]

Task 3.3

The file `container_data.csv` store data to test your program. In addition to the `container_number`, `category_of_goods` and `cargo_weight`, the first field states whether a new `ContainerNode` is being added to the `Stack` (INBOUND) or a specific `ContainerNode` is being removed from the `Stack` (OUTBOUND).

Write program code to:

- create two `Stack` instances, one designated as the `mainStack`, and the other as `spareStack`
- process each record in the `container_data.csv` file
- for INBOUND containers, invoke the appropriate methods to insert the new container into the `mainStack` and show all the containers' names in the `mainStack`
- for OUTBOUND containers, first verify whether the topmost container is the correct container to be extracted. If not, extract the topmost container from the `mainStack` and insert it into the `spareStack`. Continue this procedure until the correct container is located at the top of the `mainStack`. Remove the container and return the containers stored in the `spareStack` back into the `mainStack` until no more containers are left in the `spareStack`. Display the containers' names in both `mainStack` and `spareStack` during each operation.

[9]

Save your Jupyter Notebook for Task 3.

- 4 Dunman High School's PE department intends to develop a Web App for analysing students' NAPFA results. The csv file `napfa_results.csv` contains comma-separated values for each student and their scores across all stations (The sequence of stations is sit ups, standing broad jump, sit and reach, pull-ups, shuttle run and 2.4km run). The sample of the Web App is displayed below:

NAPFA Results

127.0.0.1:5000

Station:

☒ Sit Ups
 ☐ Standing Broad Jump
 ☐ Sit And Reach
 ☐ Pull-ups/Inclined Pull-ups
 ☐ Shuttle Run
 ☐ 2.4km Run

Order by:

☒ Ascending
 ☐ Descending

Submit

NAPFA Results:

Student Name	Gender	Sit Ups	Standing Broad Jump	Sit And Reach	Pull-ups/Inclined Pull-ups	Shuttle Run	2.4km Run
Aaron Tan	Female	26	193	50	12	12.33	11.04
Brenda Lim	Male	44	259	20	9	9.1	9.81
Cheryl Ong	Female	43	157	29	19	13.17	14.85
Daniel Lee	Female	36	182	44	13	11.47	12.7
Ethan Ng	Male	50	250	32	6	9.75	12.92
Fiona Chan	Female	42	197	44	7	10.13	12.2
Gabriel Ho	Male	40	241	35	12	11.62	11.82
Hannah Goh	Female	28	188	34	10	12.21	14.15
Isaac Tan	Male	38	230	35	13	9.94	10.29
Jolene Koh	Male	51	209	31	10	9.2	14.91
Kelvin Chua	Female	49	178	55	6	11.85	12.75
Lydia Wong	Male	33	264	29	14	11.92	11.62
Marcus Lim	Female	35	192	34	12	12.9	13.24
Nicole Tan	Female	37	161	43	7	12.08	13.24
Oscar Lee	Male	58	181	48	2	9.6	11.5
Priscilla Ng	Female	25	212	50	10	11.84	13.5
Ryan Ho	Female	42	216	38	17	11.86	15.51
Samantha Ong	Male	42	270	25	4	11.58	12.41
Timothy Chan	Male	45	209	30	5	10.94	12.64
Vanessa Lim	Male	31	185	22	10	10.89	14.48

There are 2 sets of radio buttons available for user selection, specifically Station and Order by. Upon form submission, the results in the html table will be arranged according to the chosen Station in chosen ascending or descending sequence. The sorting is to be done using a **not-in-place sorting algorithm**. Below shows the results when arranged by pull-ups in descending order. [Note: You are **NOT** permitted to use a built-in sorting function]

NAPFA Results:

Student Name	Gender	Sit Ups	Standing Broad Jump	Sit And Reach	Pull-ups/Inclined Pull-ups	Shuttle Run	2.4km Run
Cheryl Ong	Female	43	157	29	19	13.17	14.85
Ryan Ho	Female	42	216	38	17	11.86	15.51
Lydia Wong	Male	33	264	29	14	11.92	11.62
Daniel Lee	Female	36	182	44	13	11.47	12.7
Isaac Tan	Male	38	230	35	13	9.94	10.29
Aaron Tan	Female	26	193	50	12	12.33	11.04
Gabriel Ho	Male	40	241	35	12	11.62	11.82
Marcus Lim	Female	35	192	34	12	12.9	13.24
Hannah Goh	Female	28	188	34	10	12.21	14.15
Jolene Koh	Male	51	209	31	10	9.2	14.91
Priscilla Ng	Female	25	212	50	10	11.84	13.5
Vanessa Lim	Male	31	185	22	10	10.89	14.48
Brenda Lim	Male	44	259	20	9	9.1	9.81
Fiona Chan	Female	42	197	44	7	10.13	12.2
Nicole Tan	Female	37	161	43	7	12.08	13.24
Ethan Ng	Male	50	250	32	6	9.75	12.92
Kelvin Chua	Female	49	178	55	6	11.85	12.75
Timothy Chan	Male	45	209	30	5	10.94	12.64
Samantha Ong	Male	42	270	25	4	11.58	12.41
Oscar Lee	Male	58	181	48	2	9.6	11.5

Hint: Sample HTML code for radio button:

```
<input type="radio" name="gender" value="male" checked>Male<br>
<input type="radio" name="gender" value="female">Female<br>
```

<!--checked means it will be auto selected at the start-->

Task 4.1

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

[Take note that you are to read the data in the csv file `napfa_results.csv` and store it locally in the flask file.]

Save your Python program as:

```
Task_4_<your name>_<centre number>_<index number>.py
```

with any additional files/subfolders in a folder named:

```
Task_4_web_<your name>_<centre number>_<index number>
```

Run the web application.

[22]

Save the webpage output as:

```
Task_4_<your name>_<centre number>_<index number>.html
```

[1]

Save all your files for Task 4.

End of Paper

