DUNMAN HIGH SCHOOL
Preliminary Examination
Year 6

COMPUTING **9569 / 02**

Paper 2 (Lab-based) **13 Sep 2022**

3 hours

Additional materials:	Electronic version of `task1.ipynb` data file
	Electronic version of `task2.ipynb` data file
	Electronic version of `marching.csv` data file
	Electronic version of `Names.csv` data file
	Electronic version of `Results.csv` data file
	Electronic version of `contingents.json` 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 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.

The total number of marks for this paper is 100 (not including bonus questions).

Instruction to candidates:

Your program code and output for each of Task 1, 2 and 4 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>_<index number>.ipynb
```

Run all programs you create and save the output within the Jupyter Notebook files for submission unless otherwise stated.

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:

```
# Task 1.1
<Program code>
```

Task 1

Singapore's National Day celebrations mark the founding of our nation. Our National Day Parade (NDP) comprises multiple marching contingents representing the facets of our Nation and our Defence.

NDP marching contingents usually consist of individuals between 1.5 m to 1.8 m tall.

Task 1.1

Write a function `generate_heights(size)` which takes in a positive integer `size` and returns a list of `size` random integers between 150 and 180 inclusive. Test the function by calling it with size 20 as its parameter. Display the returned content.

[5]

Task 1.2

Write a function `mergesort(input_list)` which takes in any given list as `input_list`, uses merge sort to sort in ascending order, and returns the sorted list.

Write a nested function call which uses `generate_heights(size)` from Task 1.1 to generate the list and `mergesort(input_list)` to sort it. Display the returned content from the nested function call.

[6]

Parade commanders need an automated way to arrange participants by height for NDP and to inform them where to stand.

Your program will take in participant height (cm) and NRIC data and use a 2D array to generate a contingent formation similar to:

```
Row 0: [[179, 'T0663758C'], [174, 'T0849104E'], [157, 'S6254881Y'], [159, 'S7245932B'], [174, 'S7696370P'], [180, 'T2333665L']]
Row 1: [[178, 'T0223232U'], [172, 'T0230412W'], [153, 'S5334545B'], [155, 'S8547302F'], [174, 'T0434342F'], [178, 'T0343233B']]
Row 2: [[178, 'T0340912U'], [171, 'T0482309S'], [151, 'S5324433F'], [152, 'S2352434Y'], [172, 'T0320135V'], [178, 'T0281099B']]
Row 3: [[177, 'T0433322M'], [167, 'T0867787F'], [150, 'S7166399M'], [150, 'S7572379E'], [168, 'T0254885I'], [178, 'T0442131M']]
Row 4: [[175, 'S6528694I'], [161, 'T0574412P'], [150, 'S6043657K'], [150, 'S6395844J'], [164, 'S7457420P'], [175, 'T0198720F']]
Column:  0                   1                   2                   3                   4                   5
```

Note the following formation requirements:

- The shortest participants are centred and positioned nearest to the audience (Row 4), while the tallest are further back and towards the ends (columns 0 and 3)
- Each row always has an even number of participants
- There are always 5 rows; number of columns change depending on the number of participants
- The total number of soldiers for each contingent is always a multiple of 10

Task 1.3

Write a function `arrange(height_list)` which can take in a sorted list of **any** number of height-NRIC pairs and return the above 2D array format. Test `arrange(height_list)` by taking in the list `sorted_heights_nrics` (provided in task1.ipynb) and return the following 2D array

```
[[[178, 'T0281099B'], [168, 'T0254885I'], [171, 'T0482309S'], [179, 'T0663758C']],
 [[178, 'T0442131M'], [164, 'S7457420P'], [167, 'T0867787F'], [178, 'T0340912U']],
 [[175, 'S6528694I'], [159, 'S7245932B'], [161, 'T0574412P'], [175, 'T0198720F']],
 [[174, 'T0849104E'], [155, 'S8547302F'], [157, 'S6254881Y'], [174, 'S7696370P']],
 [[172, 'T0320135V'], [150, 'S7166399M'], [150, 'S7572379E'], [172, 'T0230412W']]]
```

such that each part of the list corresponds to a contingent row shown above.

The table below illustrates how `sorted_heights_nrics` is mapped onto the 2D array. The rows and columns of the table correspond to that of the contingent formation shown at the start of this task. To illustrate where each height-NRIC pair should be placed, the `sorted_heights_nrics` list index of each pair starting from `0` is placed into its corresponding row and column.

		Column			
		0	1	2	3
Row	0	[18]	[8]	[9]	[19]
	1	[16]	[6]	[7]	[17]
	2	[14]	[4]	[5]	[15]
	3	[12]	[2]	[3]	[13]
	4	[10]	[0]	[1]	[11]

Test the function by calling it and displaying the 2D array it returns.
[6]

For Task 1.4 and 1.5, you will be extending the program to allow contingent participants to query position coordinates by entering their NRIC and height.

Task 1.4

To speed up data retrieval and to respond to new data privacy requirements (only hash value of NRICs allowed to be stored, not the original NRIC), you will be creating a hash table and transferring data from a contingent 2D array into it.

For this task, you are provided with a modified array as `mod_array` in task1.ipynb. The array has been modified to include the position coordinates of each participant:

```
[[[178, 'T0281099B', [0, 0]], [168, 'T0254885I', [0, 1]], [171, 'T0482309S', [0, 2]], [179, 'T0663758C', [0, 3]]],
 [[178, 'T0442131M', [1, 0]], [164, 'S7457420P', [1, 1]], [167, 'T0867787F', [1, 2]], [178, 'T0340912U', [1, 3]]],
 [[175, 'S6528694I', [2, 0]], [159, 'S7245932B', [2, 1]], [161, 'T0574412P', [2, 2]], [175, 'T0198720F', [2, 3]]],
 [[174, 'T0849104E', [3, 0]], [155, 'S8547302F', [3, 1]], [157, 'S6254881Y', [3, 2]], [174, 'S7696370P', [3, 3]]],
 [[172, 'T0320135V', [4, 0]], [150, 'S7166399M', [4, 1]], [150, 'S7572379E', [4, 2]], [172, 'T0230412W', [4, 3]]]]
```

Initialise `hash_table` as a list representing an empty hash table of tablesize `29` with integer `-1` to indicate empty slots. Its operations are functions `hash_funct(key)`, `insert(key, data)` and `search(key, data)`.

- The hash function `hash_funct(key)` uses modulo division by its tablesize to obtain the hash value from each NRIC (digits only; exclude first and last alphabet).
- The insert and search operations should also use **linear probing** for collision resolution
- The search function should return strings informing the results of search, such as
    - `Found (with collision resolution): Row 4 Col 1`
    - `Found (no collision resolution): Row 4 Col 2`
    - `Not found (with collision resolution)`
    - `Not found (no collision resolution)`

Access and transfer the data from the `mod_array` into `hash_table`. Finally, display `hash_table` to be:

```
[-1, -1, [178, [0, 0]], [174, [3, 3]], [168, [0, 1]], [172, [4, 0]], [179,
[0, 3]], [150, [4, 1]], [172, [4, 3]], [161, [2, 2]], [171, [0, 2]], [175,
[2, 0]], [164, [1, 1]], [175, [2, 3]], [174, [3, 0]], [150, [4, 2]], [155,
[3, 1]], [178, [1, 3]], [157, [3, 2]], -1, [167, [1, 2]], [159, [2, 1]], -1,
-1, -1, -1, [178, [1, 0]], -1, -1]
```

[15]

Task 1.5

Write code to allow a contingent participant to query the program to retrieve their standing position coordinates by entering their NRIC and height. Your output should match the following for an input of S7166399M and 150:

```
Input your NRIC: S7166399M
Input your height(cm): 150
Found (with collision resolution): Row 4 Col 1
```

Run and test your code with the above example.

[1]

Task 1.6

Write test cases for `search(key, data)` and run the test cases using appropriate data.

To do this, test the function with input values by calling it using the following statement format:
`print(search(input_value1, input_value2) == expected)`

The statements should all print `True`.

[4]

Task 2

Task 2.1

The NDP parade commander needs a program to simulate the parade segment because of a recent proposal for contingent participants to simultaneously enter and leave a confined space on the parade platform.

Using a fixed array, complete the class `CircularQueue` in task2.ipynb which is a circular queue with the following methods:

- `enqueue(string)` which takes in a string and adds it to the queue or returns `"Cannot enqueue: Queue is full."` if the queue is full
- `dequeue` which removes the next item for the queue and returns the value removed

Each item enqueued represents **one participant** from a contingent.

You should **only** edit the code in task2.ipynb that is within the two above methods.

For testing purposes, you will be using a queue of `size` 5. Test your code by using the driver program provided in task2.ipynb to produce this output:

```
(1) Enqueue Participant_A, Participant_B, Participant_C, Participant_D.

Fixed array contents:
['Participant_A', 'Participant_B', 'Participant_C', 'Participant_D', None]


(2.1) Deleted value =  Participant_A
(2.2) Deleted value =  Participant_B
Fixed array contents:
['Participant_A', 'Participant_B', 'Participant_C', 'Participant_D', None]


(3) Enqueue Participant_E, Participant_F, Participant_G.

Fixed array contents:
['Participant_F', 'Participant_G', 'Participant_C', 'Participant_D',
'Participant_E']


(4) Enqueue Participant_H.

Cannot enqueue: Queue is full.
Fixed array contents:
['Participant_F', 'Participant_G', 'Participant_C', 'Participant_D',
'Participant_E']
```

[10]

Task 2.2 (Bonus question)

The parade commander wants to predict the time and name of the contingent entering the platform when the number of participants on the platform crosses a certain overload threshold of 530 people.

Write a function `simulate(filename, threshold)` which takes in a file as `filename` and a positive integer `threshold` which is the maximum number of people the platform should have. `simulate(filename, threshold)` should use `CircularQueue` from Task 2.1 and return a string with information needed by the commander.

You are provided with `marching.csv`. Its first, second, third and fourth columns correspond to each participant's NRIC, contingent name, entry-to-platform time and exit-from-platform time. Each line of `marching.csv` represents **one** person from the contingent.

Test your function with `simulate("marching.csv", 530)` to display:

```
Platform full at 6:25 pm when RSNMaritimeTrainingandDoctrineCommand is
entering.
```

[5]

Task 3

Save all your folders and files for this task in a folder named

`TASK3_<name>`

You will be creating a NDP ticketing web application `main.py` for applicants to check their NDP balloting results.

Task 3.1

You will be using a SQLite database for the program.

Create a database `ballot.db` with data from the following tables `Names` and `Results`. This data is also provided in `Names.csv` and `Results.csv`.

Results

`nric`	`group_id`	`ballot_result`
T0663758C	NULL	yes
S7829998G	4	yes
T0487877S	4	no
S8735513R	NULL	yes
T0254372E	NULL	yes
S9331280R	NULL	no
S9981511E	4	yes
S7851994I	6	no
S6450103G	6	yes

Names

`nric`	`names`
T0663758C	Farah Yusoff
S7829998G	Terry Sathyalingam
T0487877S	Lin Wee Kiat Micheal

S8735513R	Randal Khiatani
T0254372E	Chen Chi Chien
S9331280R	Natalie Lee
S9981511E	Nayeli Sng Hui Hoon
S7851994I	Cherilyn Leong
S6450103G	Emma Wijeysingha

[7]

Task 3.2

Provide SQL statements in `task3_2.sql` to

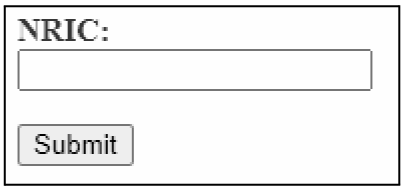
- retrieve the `ballot_result` of T0663758C
- retrieve the `name` and `ballot_result` of the other applicants who are in the same group as S9981511E (assume you do not know the `group_id` of S9981511E when writing this query)

[5]

Task 3.3

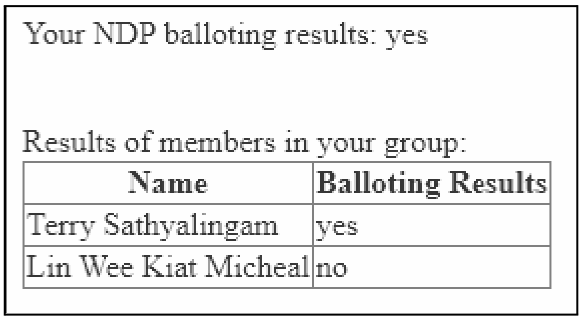
Create two HTML template files:

`home.html` will receive an NRIC as input:



Upon clicking the submit button of `home.html`, your program should use Flask to direct the browser to display the balloting results in `dashboard.html` for the NRIC entered. Invalid NRICs, NRICs which are not in the database, will be shown a page stating "Invalid NRIC".

`dashboard.html` displays the applicant's balloting results, along with a table of results of other applicants who applied to be in the same group, if any:



Your NDP balloting results: yes

Results of members in your group:

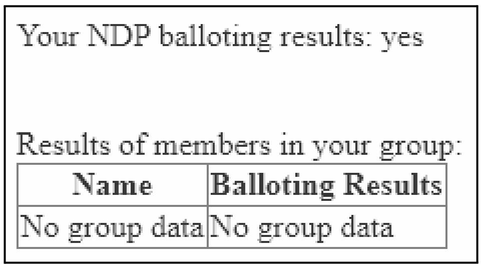
Name	Balloting Results
Terry Sathyalingam	yes
Lin Wee Kiat Micheal	no

Build the above functionalities and test the web application on a local server.
[24]

Task 3.4 (Bonus question)

Implement the following:

- The border styling of the table of dashboard.html matches the above image in Task 3.3.
- Create `styles.css` and use selectors to ensure all balloting results (group and individual) will be in red.
- For the case of an individual with no assigned grouping, the browser should display "`No group data`" in the fields:



Your NDP balloting results: yes

Results of members in your group:

Name	Balloting Results
No group data	No group data

[7]

Task 4

Task 4.1

NDP administrators help in selecting suitable applicants who want to march in NDP contingents.

Write a program to help a NDP administrator insert data into a MongoDB database `applicants` under the collection `contingents`. The data is provided for you in `contingents.json` as well as in the table below where the first row are headers for the fields.

name	medical_history	citizenship	height	suitability
Shen Rui Lin	heart issues, asthma, eczema	Singaporean	1.8	Pending
Chee Jun Ming	G6PD, heat exhaustion	PR	1.4	Pending
Yeung Jun Feng	heat exhaustion, sweaty palms	Foreigner	1.7	Pending
Deng Kai De	asthma	Singaporean		Pending

The `medical_history` field is an array of medical conditions and `height` is in metres.

Insert the above data and print out all documents in the collection `contingents`.

[6]

Task 4.2

Write code to query the database for all documents of Singaporean applicants who do not have "`heart issues`" in their medical history and who are taller than 1.5m (if they submitted a height value).

Update the `suitability` of these document(s) to "`High`".

Finally, print out **all** documents in `contingents`.

[5]