NANYANG JUNIOR COLLEGE
TEMASEK JUNIOR COLLEGE
VICTORIA JUNIOR COLLEGE

JC2 PRELIMINARY EXAMINATIONS

Higher 2

COMPUTING 9569/02

Paper 2 (Lab-based) **24th August 2022**

3 Hours

Additional Materials:	Removable storage device
	Electronic version of `NUMBERS.txt` data file
	Electronic version of `libraryitems.txt` data file
	Electronic version of `Task2_timedelta.py` file
	Electronic version of `IDs.txt` data file
	Electronic version of `Task4.db` 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.

This document consists of **8** printed pages, **2** blank pages and **1** insert.

[Turn over

Instruction to candidates:

Your program code and output for each of Task 1, 2 and 3 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>_<centre number>_<index number>.ipynb`

1 The task is to:

- read a sequence of numbers from a file
- sort the numbers using a merge sort algorithm
- search for a given number within the sorted data using binary search.

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(list_of_strings)` that:

- takes a list of strings, `list_of_strings`, where each string is a natural number
- sorts the list in ascending numerical order, using a merge sort algorithm
- returns the sorted list of strings. [7]

Use the list `['11', '101', '10', '1', '111', '1100', '100']` to test your function.

It should return the result `['1', '10', '11', '100', '101', '111', '1100']`. [1]

Task 1.2

Write a function `task1_2(filename)` that:

- takes a string `filename` which represents the name of a text file
- reads in the contents of the text file as a list of strings
- uses the `task1_1` function to sort the strings
- returns the sorted list of strings. [4]

Task 1.3

Write a function `task1_3(list_of_sorted_strings, num)` that:

- accepts two parameters:
  - o `list_of_sorted_strings`, the list of sorted data
  - o `num`, a string representing the number that is being searched for
- implements the Binary Search algorithm to return the index of the first element in `list_of_sorted_strings` that matches `num`
- returns `-1` if no element matching `num` is found. [5]

Call your function `task1_3` with the contents of the file `NUMBERS.txt`, printing the returned values, using the following statements:

```
print(task1_3(task1_2('NUMBERS.txt'), '667'))
print(task1_3(task1_2('NUMBERS.txt'), '130'))
```
[2]

Save your Jupyter notebook for Task 1.

2 A college decides to digitise its library's loan system. The college library currently has two types of items for loan: books and compact discs (CDs). The college intends to implement the college's library's loan system using object-oriented programming (OOP).

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

Registered library users have a `user_id` which they use for all transactions. Library items can be loaned for three weeks.

The class `LibraryItem` will store the following data:

- `item_id` – stored as a string
- `title` – stored as a string
- `loaned_by` – the `user_id` of the user whom the item is loaned to, or an empty string if the item is not on loan
- `due_date` – stored as a date with the format YYYY-MM-DD

The class has four methods defined on it:

- `is_on_loan()` – returns False if `loaned_by` is empty, otherwise returns True
- `return_item()` – sets `loaned_by` to empty string
- `loan_to(user_id)` – returns a Boolean value to indicate whether the user has been loaned a library item successfully
  A library item that is already on loan cannot be loaned out again
- `print_details()`
  - o display `title` and `due_date` of the `LibraryItem` in separate lines.
  - o display `user_id` of the user(s) loaning the item, if any. E.g.

```
Title: Thinking with type
Loaned by: S1111111H
Due date: 2022-09-03
```

The `timedelta` library is built into Python and can be used to perform arithmetic operations on dates. Example code is shown in `Task2_timedelta.py`.

Write program code in Python to define the base class `LibraryItem.` [9]

Task 2.2

The `Book` class inherits from `LibraryItem`, such that:

- the following extra information is recorded:
  - `author` – stored as a string
  - `category` – stored as a string
  - `reserved_by` – the `user_id` of the user whom the item is reserved by, or an empty string if the item is not reserved
- they may be reserved, and reservation is handled through the following methods:
  - `is_reserved()` – returns False if `reserved_by` is empty, otherwise returns True
  - `release()` – sets `reserved_by` to empty string
  - `reserve_for(user_id)` – returns a Boolean value to indicate whether the user has reserved a library item successfully
    A library item that is already reserved cannot be reserved again.
- `print_details()` should display:
  - two additional attributes, `author` and `category`
  - `user_id` of the user(s) reserving the item, if any.

The `CompactDisc` class inherits from `LibraryItem`, such that:

- the following extra information is recorded
  - `artist` – stored as a string
  - `genre` – stored as a string
- `print_details()` should display two additional attributes, `genre` and `artist`.

Write program code to define the `Book` and `CD` subclasses. [6]

Task 2.3

The text file, `libraryitems.txt,` contains information for several books and CDs. Each row is a comma-separated list of information for a book or a CD. The item IDs for books begin with a B, while those for CDs begin with a C.

The information of a book is given in the following order:
```
item_id, title, author, category
```

The information of a CD is given in the following order:
```
item_id, title, artist, genre
```

Write program code to:

- read in the information from the text file, `libraryitems.txt`
- create an instance of the appropriate class for each library item
- display the information of each instance
- store the library items as a list of object instances. [5]

Task 2.4

User ID `S1111111H` borrows all the library items in the text file, `libraryitems.txt`.
User ID `S1222222D` reserves all the books that `S1111111H` has borrowed.

Write program code to carry out the above steps. Print out the details of all library items. [4]

Save your Jupyter notebook for Task 2.

3 The file `IDs.txt` contains 5 membership numbers and member names for a club.

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

The IT administrator of the club wishes to store the membership details in a hash table using the following hash algorithm:

`address = k MOD 31`, where `k` is the membership number

Write program code to implement the hash table described above. Your code should do the following:

- create a hash table with 31 addresses
- implement the hash algorithm as described above
- add a new membership record to the hash table, via a function or method `add()`
  - This function should also check if the address is occupied
  - If the address is already occupied, the membership number is not added and an appropriate message should be displayed instead
- display the contents of the hash table, via a function or method `display()`. [10]

Task 3.2

Amend your code from **Task 3.1** to implement lookup for the hash table. The function/method `search()` should:

- take in an integer value `membershipNo`
- return the member name associated with `membershipNo`
- if `membershipNo` is not found, an appropriate message should be displayed. [3]

Task 3.3

The current hash table implementation will not generate unique addresses. For example, membership numbers 263 and 666 will both generate the address value 15.

For such collision of membership numbers, a second hash algorithm can be used:

`address = (2^k) MOD 31`, where `k` is the membership number

Amend your code from **Task 3.2** to implement rehashing with the above hash algorithm in case of collisions of membership numbers. [6]

Test your program by inserting the membership records from the text file `IDs.txt` into the hash table sequentially. Display the contents of the hash table after all records are inserted. [5]

Save your Jupyter notebook for Task 3.

BLANK PAGE

[Turn over

4 A security guard post uses a database to store entry and exit records for people entering and exiting a school. People with a teaching role at the school are referred to as 'staff', people enrolled in lessons at the school are referred to as 'students', and all others are referred to as 'visitors'.

The school wishes to create a web page to add records to this database and summarise its data. Some visitor data is provided in a database, `Task4.db`, provided with this question.

The database has a `Record` table with the following columns:

- `id` — the primary key of the record
- `Date` — date of entry, in YYYY-MM-DD format
- `Time` — time of entry, in 24-hour format (e.g. `2359` for 11:59pm)
- `Type` — 'entry' or 'exit'
- `visitorId` — primary key of the record in the `Person` table

The database also has a `Person` table with the following columns:

- `id` — the primary key of the record
- `Role` — 'Staff', 'Student', or 'Visitor'
- `Name` — The person's full name.

Task 4.1

Write a Python program and the necessary files to create a web application. The web application offers the following menu options:

Latecomers
Add entry/exit record

Save your program code as

`Task4_1_<your name>_<centre number>_<index_number>.py`

With any additional files/subfolders as needed in a folder named

`Task4_1_<your name>_<centre number>_<index_ number>`

Run the web application and save the output of the program as

`Task4_1_<your name>_<centre number>_<index number>.html` [5]

Task 4.2

Write an SQL query that shows, for students entering the school after `0730`:

- name of student
- entry time
- sorted by date and time
- in ascending order.

The data should be displayed in a web page that is accessed from the “Latecomers” menu option from **Task 4.1**.

Save all your SQL code as

`TASK4_2_<your name>_<centre number>_<index number>.sql`

With any additional files/subfolders as needed in a folder named

`Task4_2_<your name>_<centre number>_<index_ number>` [6]

Run the web application and save the output of the program as

`Task4_2_<your name>_<centre number>_<index number>.html` [2]

Task 4.3

The data from the database, `Task4.db`, is to be used to implement a visitor entry/exit form in a web browser.

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

- displays a web form
- enables the user to enter visitor ID and direction (`entry` or `exit`)
- stores the record in the database
- shows a success or error message alongside the entry details.

The web application should be accessed from the appropriate menu option from **Task 4.1**.

Save your program as

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

with any additional files / sub-folders as needed in a folder named

`TASK4_3_<your name>_<centre number>_<index number>` [11]

Run the web application and add the following entries using the web form:

- visitor ID: `1`, Date: `2022-10-05`, Time: `0722`, Type: `entry`
- visitor ID: `A`, Date: `2022-10-05`, Time: `0722`, Type: `entry`
- visitor ID: `7`, Date: `2022-10-05`, Time: `0722`, Type: `entry`

Save the output of the program as

`Task4_3_<your name>_<centre number>_<index number>_1.html`

`Task4_3_<your name>_<centre number>_<index number>_2.html`

`Task4_3_<your name>_<centre number>_<index number>_3.html` [3]