H

TEMASEK JUNIOR COLLEGE

8352

2023 JC2 MID YEAR EXAMINATION PC013 EXAM 0702

Higher 2

TEMASEK JUNIOR COLLEGE

COMPUTING

9569/02

Paper 2 (Lab-based)

4 July 2023

3 hours

Additional Materials: Removable storage device
Electronic version of `HASH.csv` data file
Electronic version of `Task3_timedelta.py` file
Electronic version of `Task3_strptime.py` file
Electronic version of `Task3_strftime.py` file
Electronic version of `MIMS.txt` data file
Electronic version of `Task4.db` file
Electronic version of `EQUIPMENT.txt` data file
Electronic version of `UNITS.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 and 2 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`

1 Name your Jupyter Notebook as:

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

The task is to write a program to perform sort and search operations on a list of randomly generated unique integers.

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:

```
[1] :   # Task 1.1
        Program code

        Output:
```

Task 1.1

Write code for a procedure `task1_1()` to display a menu with the following options:

```
1 - Bubble Sort
2 - Insertion Sort
3 - Quick Sort
4 - Merge Sort
5 - Linear Search
6 - Binary Search
7 - End
```

[1]

Task 1.2

Write a function `task1_2(arr)` that:

- takes in an array of unique integers, `arr`
- sorts the integers in `arr` in ascending order using an iterative bubble sort algorithm
- displays and returns the sorted array of unique integers. [3]

Task 1.3

Write a function `task1_3(arr)` that:

- takes in an array of unique integers, `arr`
- sorts the integers in `arr` in ascending order using an iterative insertion sort algorithm
- displays and returns the sorted array of unique integers. [3]

Task 1.4

Write a function `task1_4(arr)` that:

- takes in an array of unique integers, `arr`
- sorts the integers in `arr` in ascending order using an out-of-place recursive quicksort algorithm
- displays and returns the sorted array of unique integers. [4]

Task 1.5

Write a function `task1_5(arr)` that:

- takes in an array of unique integers, `arr`
- sorts the integers in `arr` in ascending order using a recursive merge sort algorithm
- displays and returns the sorted array of unique integers. [6]

Task 1.6

Write a function `task1_6(target, arr)` that:

- takes in an array of unique integers, `arr`
- searches the `arr` for the value `target` using either an iterative or a recursive linear search algorithm
- returns the position of `target` in `arr` or `-1` if it does not exist.

You are only required to present either an iterative or a recursive linear search algorithm but not both. [2]

Task 1.7

Write a function `task1_7(target, arr)` that:

- takes in an array of unique integers, `arr`
- searches the `arr` for the value `target` using either an iterative or a recursive binary search algorithm
- returns the position of `target` in `arr` or `-1` if it does not exist.

You are only required to present either an iterative or a recursive binary search algorithm but not both. [5]

Task 1.8

Write a function `task1_8(n, minimum, maximum)` that:

- takes in three positive integer parameters:
  - `n`, the number of unique random integers to generate
  - `minimum`, the smallest value a randomly generated integer can take
  - `maximum`, the largest value a randomly generated integer can take
- generates a list of `n` unique random integers from the range `minimum` to `maximum`
- displays and returns the list of randomly generated unique integers. [3]

Task 1.9

Write a program to implement the menu in **Task 1.1** using the functions written in **Task 1.2** through **Task 1.8**.

Your program should:

- generate and display a list of unique random integers
- prompt the user to select one of the sort or search operations in the menu
- ensure all requirements, if any, for the selected operation to run correctly is/are satisfied
- output an error message on all unmet requirements, if any, for the selected operation
- allow the user to perform all necessary operations to meet all unmet requirements
- perform the selected operation and display the result [4]

Save your Jupyter Notebook for Task 1.

2 Name your Jupyter Notebook as:

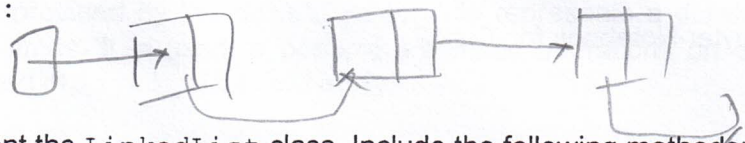
`TASK2_<your name>_<centre number>_<index number>.ipynb`

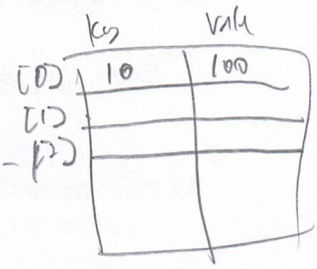
The task is to implement a hash table that handles collisions using a chaining algorithm that makes use of a linked list data structure.

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:

```
[1] :   # Task 2.1
        Program code
```

Output:





Task 2.1

Write code to implement the `LinkedList` class. Include the following methods:

- suitable constructor method for instantiating an empty `LinkedList` object.
- `insert(key, value)` which takes in an integer `key` and a string `value` and inserts them at the beginning (head) of the list
- `search(key)` which takes in an integer `key` and returns the corresponding `value` and the `position` of `key` in the linked list as a string; if the `key` was not present, return `None`
  - the first `position` is `0`, which is the beginning (head) of the list
  - the return string should be in the format "`value` found at `position`".
- `display()` which outputs a list of `key-value` tuples. [8]

Task 2.2

Write code to implement the `HashTable` class. Include the following methods:

- suitable constructor method for instantiating an empty `HashTable` object of size `n`, where each slot stores an empty `LinkedList` object
- `hash_algorithm(key)` that:
  - takes in an integer `key`
  - implements the hashing algorithm `key MOD hash table size`
  - returns the index of the hash table slot where the `key-value` pair should be inserted
- `hash_insert(key, value)` which inserts the `key-value` pair into the hash table
- `hash_search(key)` which searches for location of `key` and its corresponding `value`
- `hash_display()` which outputs all the contents of the hash table in rows, with the contents of each slot occupying a single row. [10]

Task 2.3

The comma-separated-values (csv) file `HASH.csv` contains a set of 13 records for testing your `HashTable` class implementation.

Each row contains one record stored in the following format:

```
key,value
```

Write program code to:

- insert all data in `HASH.csv` into a hash table of size 25
- search for the records with `key` 34 and `key` 59
- display the contents of the hash table after all records have been inserted. [4]

Save your Jupyter Notebook for Task 2.

3 A student will be locked out of the Student MIMS account once the password has exceeded the maximum validity duration of 90 days. This has an impact on daily learning as ICT provisions such as the SWN@SSOE WIFI and Student ICON services cannot be accessed.

To ensure that all students reset their MIMS account passwords timely, the ICT team of a particular school has decided to pilot a tracking system for the validity of students' MIMS account passwords. This system will generate a reminder email to any student whose MIMS account password is expiring within the next 10 days, instructing them to reset it before the expiry date.

The task is to create the initial prototype of this tracking system for the first phase of the pilot.

Task 3.1

The `timedelta` class provided by the `datetime` module represents a duration or difference between two dates or times. It is used to perform arithmetic operations on dates and times. Example code is provided in `Task3_timedelta.py`.

The `strptime()` method provided by the `datetime` module converts a string representation of a date and time into a `datetime` object. Example code is provided in `Task3_strptime.py`.

The `strftime()` method provided by the `datetime` module converts a `datetime` object into a string representation. Example code is provided in `Task3_strftime.py`.

Using the `timedelta` class, the `strptime()` method and the `strftime()` method, or otherwise, write a function `task3_1(strdate, n)` that:

- takes in two parameters:
  - `strdate`, a string representation of a date in the format `YYYYMMDD`
  - `n`, a positive integer representing the number of days after `strdate`
- determines the resultant date `n` days after `strdate`
- returns the resultant date as a string in the format `YYYYMMDD`. [3]

Use your function to determine the expiry date of the MIMS account password for a student who has just onboarded the MIMS system today. [1]

Save your code as
`Task3_1_<your name>_<centre number>_<index number>.py`

Task 3.2

The file `MIMS.txt` contains the records of students in a particular class, including the dates of onboarding the MIMS system and the last date of password reset.

Each line in `MIMS.txt` is a record of one student in the following format:

```
_id,name,account,date_onboarded,date_last_reset
```

where:

- `_id` is the unique identification number assigned to the student
- `name` is the name of the student
- `account` is the MIMS account of the student
- `date_onboarded` is a text string in the format `YYYYMMDD`, representing the date which the student have onboarded the MIMS system
- `date_last_reset` is a text string in the format `YYYYMMDD`, representing the date the last password reset was performed.

Write program code to insert the following data for each student into a NoSQL database `mims` under the collection `password`:

- `_id`, `name`, `account`, `date_onboarded`, `date_last_reset`, all of which can be obtained from each record in `MIMS.txt`
- `date_next_reset`, a text string in the format `YYYYMMDD`, which can be calculated to be 90 days from the date of last reset using the `task3_1(strdate, n)` function in **Task 3.1**.

You will need to copy and paste the code for the function `task3_1(strdate, n)` from **Task 3.1** to the front of your program code. [5]

Save your code as
`Task3_2_<your name>_<centre number>_<index number>.py`

Task 3.3

Once a student resets the MIMS account password, the record in the `mims` database should be updated.

Write code for a procedure `task3_3(_id, strdate)` that:

- takes in two parameters
  - `_id`, the unique identification number of the student
  - `strdate`, a string in the format `YYYYMMDD`, representing the date the student resets the MIMS account password
- uses the function `task3_1(strdate, n)` in **Task 3.1** to obtain the expiry date of the new MIMS account password
- updates the values of `date_last_reset` and `date_next_reset` for the student's record in the `mims` database.

You will need to copy and paste the code for the function `task3_1(strdate, n)` from **Task 3.1** to the front of your program code. [3]

The student with `_id = 1` has reset the MIMS account password today.

Write additional code to use your procedure to update the student's records in the `mims` database. [1]

Save your code as
`Task3_3_<your name>_<centre number>_<index number>.py`

Task 3.4

Every working day, the ICT team will activate the tracking system once to generate email reminders for students whose MIMS account passwords are expiring within the next 10 days. This process is expected to be automated in the second phase of the pilot but will not be considered at this stage.

The text for the email reminder is as such:

```
Dear <name of student>,

Your MIMS account password will be expiring soon.  Please perform a
self-reset of your MIMS account password by <date_next_reset>.
```

Write code for a procedure `task3_4(strdate)` that:

- takes in a parameter `strdate`, a string in the format `YYYYMMDD`, representing the date of the working day for which the tracking system is activated
- identifies all students in the `mims` database whose MIMS account password will expire within 10 days from `strdate`
- for each student being identified, generate the text for the email reminder
- write the text for the email reminder into the file `EMAIL_REMINDER.txt`, leaving a line after the end of the text for the next email reminder to be written. [5]

Generate the text for the email reminders for all students whose MIMS account passwords are expire within 10 days from today. [1]

Save your code as
`Task3_4_<your name>_<centre number>_<index number>.py`

4 The Infocomm Club of a particular school wants to design a web-based inventory system to store the records of all its equipment in a relational database. The tables for the database have been created and stored in `Task4.db`.

There are two tables in the database:

- `equipment(equipmentID, model, price)`
- `units(unitID, equipmentID)`

The Infocomm Club currently stores its information in two text files, `EQUIPMENT.txt` and `UNITS.txt` and would like to reduce duplication of information by storing all the data into the relational database `Task4.db`.

Each line in `EQUIPMENT.txt` contains information regarding an equipment in the following format:

```
equipmentID,model,price
```

Each line in `UNITS.txt` contains information regarding one unit of an equipment in the following format:

```
unitID,equipmentID,model,price
```

The task is to design the web-based inventory system.

Task 4.1

Write a Python program to insert all information from the files into `Task4.db`.

Run the program. [4]

Save your program as
`Task4_1<your name>_<centre number>_<index number>.py`

Task 4.2

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

```
Insert a new equipment record
Display all equipment
```

Save your program as
`Task4_2<your name>_<centre number>_<index number>.py`

with any additional files/subfolders as needed in a folder named
`Task4_2<your name>_<centre number>_<index number>`

Run the web application and save the output of the program as
`Task4_2<your name>_<centre number>_<index number>.html` [3]

Task 4.3

The Infocomm Club would like to extend the web application to:

- receive `equipmentID`, `model`, `price`, and the number of units for a new equipment to be inserted into `Task4.db`
- assign a `unitID` to each unit of the equipment
- insert the data into the respective tables of `Task4.db`.

The `unitID` is a three-digit string starting with `001`. For example, if there are three units of an equipment, they would have values of the `unitID` to be `001`, `002` and `003` respectively.

Write program code for the extension such that the web application interface for receiving the required information should be accessed from the appropriate menu option in **Task 4.2**.

You will need to copy and paste your program code from **Task 4.2** to the front of your program code. [6]

Test your program by inserting 5 units an equipment with the following details into `Task4.db`:

```
equipmentID: 124765
model: Canon EOS R10
price: 1749
```
[1]

Save your program code as
`Task4_3<your name>_<centre number>_<index number>.py`

with any additional files/subfolders as needed in a folder named
`Task4_3<your name>_<centre number>_<index number>`

Task 4.4

The Infocomm Club would like the web application to display information for all the equipment in a table.

Write program code for the extension to show the `equipmentID`, `model`, `price` and the number of units of that equipment in a table. The results should be shown in alphabetical order by model. The web application interface for displaying the information should be accessed from the appropriate menu option in **Task 4.2**.

You will need to copy and paste your program code from **Task 4.3** (including that of **Task 4.2** at the front of the program code) to the front of your program code. [6]

Save your program as
`Task4_4<your name>_<centre number>_<index number>.py`

with any additional files/subfolders as needed in a folder named
`Task4_4<your name>_<centre number>_<index number>`

Run the web application and save the output of the program as
`Task4_4<your name>_<centre number>_<index number>.html` [2]

- End of Paper -