H

TEMASEK JUNIOR COLLEGE

2023 JC2 PRELIMINARY EXAMINATION

Higher 2

TEMASEK JUNIOR COLLEGE

COMPUTING

9569/02

Paper 2 (Lab-based)

14 August 2023

3 hours

Additional Materials:	Removable storage device
	Electronic version of `CPU-PROCESSES.csv` data file
	Electronic version of `SPORTS-FACILITIES.txt` 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.

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 freight (shipping) container identification is covered by the ISO 6346 standard. Under ISO 6346, each container is labelled with an 11-character code (four letters + seven digits) in which the last digit is a "check" digit that is computed from the leftmost 4 letters and 6 digits, according to a fixed rule. For example, in MSKU3881107, the final "7" is the check digit.

The check digit is calculated by applying the following algorithm, on the left-most 4 letters and 6 digits of a freight container identification:

1. The four letters are replaced with corresponding numbers. The letters A through Z correspond to the numbers 10 through 38, with all multiples of 11 skipped.
2. The 10 numbers are multiplied by successive powers of two, starting from 2^0 to 2^9, and added together.
3. Take the remainder when dividing by 11, and if the result is 10, it is taken to be zero.

For example, given the 4 characters and 6 digits MSKU388110:

Step 1: 24 30 21 32 3 8 8 1 1 0

Step 2: $24\times2^0 + 30\times2^1 + 21\times2^2 + 32\times2^3 + 3\times2^4 + 8\times2^5 + 8\times2^6 + 1\times2^7 + 1\times2^8 + 0\times2^9 = 1624$

Step 3: 1624 / 11 = 147 remainder 7, hence check digit = 7

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(input_value)` that returns a string.

The function should:

- validate that the parameter `input_value` is a string of 4 letters followed by 7 digits
- return `True` if the above rule is met, otherwise return `False`. [5]

Task 1.2

Create four tests to test your function. Display the input value used for each test, and its result.

Your four input values should be:

- a string containing only integers
- a string containing only letters
- a valid string
- a string of incorrect length.

Test your function with your four input values by calling it using the following statement:

```
print(taskl_l(input_value))
```
[4]

Task 1.3

Write a function `taskl_3(input_value)` that returns the calculated check digit.

The function should:

- take in an `input_value` containing the left-most 4 letters and 6 digits of the 11-character code
- calculate the check digit
- return the calculated check digit. [3]

Task 1.4

A full 11-character freight container identification can be validated by removing the check digit, calculating the check digit from the remaining 10 alphanumeric characters and comparing it to the removed digit.

Write a function `task1_4(input_value)` that returns a Boolean indicating whether the check digit is valid. The function should:

- validate the parameter `input_value` by calling your function from **Task 1.1**
- return `False` if `input_value` is invalid
- remove the rightmost digit
- use your function from **Task 1.3** to calculate the check digit from the remaining string
- return `True` if the calculated check digit is the same as the check digit in the identification code and `False` otherwise. [3]

Test your function with the following statements:

```
print(task1_4('PONU2079674'))

print(task1_4('XONU2079674'))
```
[2]

Task 1.5

The rule specified by ISO 6346 for computing the check digit is designed so that accidental changes or misreading of a single digit in a code can by identified as it will also change the check digit. Hence, most such errors can be caught easily using the check digit. However, the check digit can sometimes fail to detect a change in the code.

Create two test cases such that one test case is different from the other test case by a single character yet both test cases produce the same check digit.

Test your function with the following statements:

```
print(task1_4(test_case_1))
```

```
print(task1_4(test_case_2))
```

Both statements should return `True`. [2]

Save your Jupyter notebook for Task 1.

2 The file `SPORTS-FACILITIES.txt` contains the total number of bookings for different sports facilities managed by Sport Singapore for the year 2022. Each line contains comma-delimited data that shows the type of facility and the corresponding number of bookings for the facility in 2022 in the following format:

```
Facility,Number of Bookings
```

The task is to read the data from the file `SPORTS-FACILITIES.txt`, use various sorting algorithms to display a sorted version of the data, and then compare the efficiencies of the sorting algorithms implemented.

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

One method of sorting data is insertion sort.

Write program code to:

- read the facilities booking data from `SPORTS-FACILITIES.txt`
- implement an insertion sort algorithm to sort the data in descending order based on the number of bookings
- display the sorted data in rows and columns where:
    - o the first column of each row contains data on one type of facility
    - o the second column of each row contains the corresponding number of bookings. [7]

Task 2.2

Another method of sorting data is quicksort.

Write program code to:

- read the facilities booking data from `SPORTS-FACILITIES.txt`
- implement a quicksort algorithm to sort the data in descending order based on the number of bookings
- display the sorted data in rows and columns where:
    - o the first column of each row contains data on one type of facility
    - o the second column of each row contains the corresponding number of bookings. [7]

Task 2.3

The sorting algorithms implemented in **Task 2.1** and **Task 2.2** can be modified to track the number of comparisons made during each sorting process. The number of comparisons can then be used to identify which sorting algorithm is more efficient for this particular instance.

Copy and paste your program codes in **Task 2.1** and **Task 2.2** into a new cell in your Jupyter Notebook.

Modify your program code to:
- track the number of comparisons made by the sorting algorithm implemented in each task
- display these numbers
- determine which sorting process is faster for this instance
- output an appropriate conclusion. [5]

Save your Jupyter notebook for Task 2.

3 A scheduler is a queue used to ensure that processes in a computer are completed in timely fashion, according to a scheduling algorithm.

One such algorithm is the first-come-first-serve algorithm (FCFS), in which each process in the queue is executed until completion before moving on to the next task.

Another algorithm is the round-robin scheduling algorithm (RR), in which each process is given a fixed interval of time to execute. The process is

- removed from the queue immediately upon completion,
- paused and moved from the front to the rear of the queue if not yet completed at the end of the interval.

The next process that is now at the front of the queue is then given the same interval of time to execute.

Each process is represented using Object-Oriented Programming (OOP) by a `Process` class, which contains two properties:

- `name` is a label for the process
- `remaining` is the number of seconds left for the process to complete.

The class `Process` contains the following methods:

- `execute(t)` executes the process for `t` seconds. The remaining time is reduced by `t` seconds
- `is_completed()` returns a Boolean representing whether the process is completed. A process is considered completed if the remaining time is zero or less.

The scheduler uses a circular queue to manage processes. The `CircularQueue` class contains the following properties:

- `items` is an array representing the items in the queue
- `front` refers to the index of the first item in the queue
- `rear` refers to the index of the last item in the queue.

The `CircularQueue` class contains the following methods:

- `is_empty()` returns `True` if queue is empty, or `False` otherwise
- `is_full()` returns `True` if queue is full, or `False` otherwise
- `enqueue(process)` adds the process to the queue, or displays an appropriate message if the queue is full
- `dequeue()` returns a process, removing it from the queue, or displays an appropriate message if the queue is empty
- `display()` outputs processes from the front to the rear of the queue, or displays an appropriate message if the queue is empty.

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

Write program code to implement the `Process` class. [4]

Task 3.2

Write program code to implement the `CircularQueue` class. [12]

Task 3.3

The `FCFSScheduler` and `RRScheduler` classes inherit from the `CircularQueue` class, and implement the FCFS and RR scheduling algorithms respectively. Both scheduler classes contain the following method, in addition to the methods implemented by `CircularQueue`:

- `start()` causes the scheduler to run in a loop, doing the following in each iteration:
  - o dequeue the first process from the queue
  - o execute the process according to the scheduling algorithm
  - o if the process is completed, display "`<Process name> completed`", substituting `<Process name>` with the name of the process. The process is not added back into the queue
  - o if the process is incomplete, display "`<Process name>: <t> seconds remaining`", substituting `<Process name>` with the name of the process and `<t>` with the remaining time for the process. The process is enqueued into the scheduler again
  - o if the scheduler is empty, terminate the loop.

Write program code to implement the `FCFSScheduler` and `RRScheduler` classes. You may use a time interval of 0.5 seconds for the execution of processes. [8]

Task 3.4

The file `CPU-PROCESSES.csv` contains data of ten CPU processes in comma-delimited format, with a header row.

Each subsequent row contains data of a CPU process in the format:

```
<Process name><Time required for completion>
```

Write program code to:

- initialise a scheduler of the appropriate type
- read the data of the processes from `CPU-PROCESSES.csv`
- enqueue all the processes into the scheduler
- start the scheduler.

In separate cells, show the output from your code clearly for both schedulers. [4]

Save your Jupyter notebook for Task 3.

4 A sports club uses the database, `Task4.db`, to keep track of its activities.

The database has three tables to store information on:

- the courses available
- the members who come to their club
- the certificates issued for completed courses.

The descriptions for the tables are as follows:

- `Course (CourseCode, Description, Fee)`
- `Member (MemberId, FirstName, LastName, Phone)`
- `Certificate (MemberId, CourseCode, IssueDate, Status)`

Task 4.1

As part of an upgrading programme, the club decides to extend its functionality to the web.

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

Menu
Delete Erroneous Certs
VIP Voucher Recipients
Add Certificate 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` [4]

Task 4.2

In accordance with their regulations, certificate records deemed erroneous are flagged with a 3-digit status code of 404.

Write an SQL query that removes the erroneous data records in the certificates table and counts the number of records removed.

The resulting number of records removed should be labelled clearly and shown on a web page, accessed from the `Delete Erroneous Certs` 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>` [4]

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

`Task4_2_<your name>_<centre number>_<index number>.html` [1]

Task 4.3

As part of a member incentive, members who were issued course certificates after January 2023 are entitled to a VIP voucher.

Write an SQL query that shows, for members who are VIP voucher recipients, their:

- first name and last name
- phone number
- course description
- certificate issue date
- sorted by last name and first name
- in ascending order.

The resulting member details should be shown on a web page in a table, accessed from the appropriate menu option from **Task 4.1**.

Save all your SQL code as

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

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

`Task4_3_<your name>_<centre number>_<index_ number>` [7]

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

`Task4_3_<your name>_<centre number>_<index number>.html` [1]

Task 4.4

The data from the database, `Task4.db`, is to be used to implement a certificate record entry 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 the following information:
  - member ID
  - course code
  - issue date
- sets the status code to 200 by default
- 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_4_<your name>_<centre number>_<index number>.py`

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

`TASK4_4_<your name>_<centre number>_<index number>` [9]

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

- member ID: `145543`, course code: `CR0001`, issue date: `2023-05-27`
- member ID: `151179`, course code: `CR0001`, issue date: `2023-05-27`.

Save the outputs of the program as

`Task4_4_<your name>_<centre number>_<index number>_1.html`

`Task4_4_<your name>_<centre number>_<index number>_2.html` [2]

-- END OF PAPER --