HWA CHONG INSTITUTION C2 PRELIMINARY EXAMINATION 2022

COMPUTING Higher 2 Paper 2 (9569 / 02)

23 AUG 2022

1400 -- 1700 hrs

Additional Materials:

Electronic version of MINEFIELD.txt data file

Electronic version of booklist.csv data file

Electronic version of newbooks.csv data file

Electronic version of students.db 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 **4** 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.

Instructions to candidates:

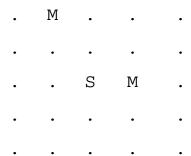
Your program code and output for each of Task 1, 2 and 3 should be downloaded in a single .ipynb file. For example, your program code and output for Task 1 should be downloaded as TASK1_<your name>_<center number>_<index number>.ipynb

1. Design a computer program to simulate the survival game of a soldier inside a minefield.

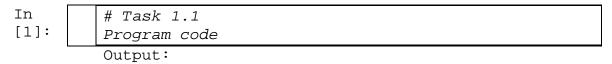
The minefield is to be represented on the screen by a nxn square grid where n is an odd integer. Each cell of the grid is represented by an x-coordinate and a y-coordinate. The top left cell display has x = 0 and y = 0.

A cell without any mine is displayed as '•' and a cell with mine is displayed as 'M'. The soldier is displayed as 'S', standing at the centre of the grid at the beginning of the game.

For example, a minefield of 5x5 with mines at (0, 1), (2, 3) is displayed as



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

The text file MINEFIELD. TXT provides the size of the minefield and the coordinates of the mines. The first line of the file is the value of n and the subsequent lines are formatted as

```
<x-coordiante>,<y-coordiante>
```

Write program code to

- read the text file
- create the data structure to represent the grid
- display the minefield with soldier and the mines in it

[5]

Task 1.2

The soldier attempts to walk out of the minefield. Each time he sets off he is equally likely to go UP, DOWN, LEFT, or RIGHT. When he moves onto a safe cell without mine, the cell is displayed as 'P' instead. He loses the game once stepping onto a mine. Otherwise, when he walks safely to any of the four boundaries, he wins the game.

Using the data structure created from Task 1.1, write program code to simulate the game. When the game ends, display the steps he takes, the grid and the outcome. Below is a sample run.

```
STEPS: LEFT UP LEFT LEFT
. . M . . . .
. . . M . . .
P P P . . . M
. . P S M . .
. . . . . .
. . . . . .
WIN! You walked to the boundary!
```

[10]

Save your Jupyter notebook for Task 1.

2. A school library club polled its members and created a list of 18 recommended classics. The list is saved in a file named booklist.csv stating the book title, author and year of publication.

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 2.1
[1]: Program code
Output:
```

Task 2.1

Write a function read_csv(filename) which

- reads a csv file, filename
- stores the records in an array with 3 columns title, author and year
- title, author and year all stored as strings
- returns the array

Use the following code to test your function.
books_array = read_csv("booklist.csv")
print(len(books_array))
print(books_array)
[4]

Task 2.2

As the data is not suitably ordered, a sort algorithm is required.

Write a function to implement a bubble sort algorithm bubble(array, sort_key) which sorts array by the sort_key provided.

- The sort_key should be one of the values "title", "author" or "year"
- The function will return the sorted array in ascending order
- The function will give a return of -1 if an invalid sort key is provided

Use the following code to test your function.
print(bubble(books_array, "title"))

print(bubble(books_array, "ISBN")) [6]

Task 2.3

When the same algorithm was used on the entire library catalogue, it was deemed to be too slow.

Write a function to implement a merge sort algorithm merge (array, sort_key) which sorts array by the sort_key provided.

- The sort_key should be one of the values "title", "author" or "year"
- The function will return the sorted array in ascending order
- The function will give a return of -1 if an invalid sort_key is provided

Use the following code to test your function.

```
print(merge(books_array, "author"))
print(merge(books_array, "year")) [7]
```

Task 2.4

Write a function reverse(array) which reverses the order of the array, without the use of any built-in functions.

```
Use the following code to test your function.

print(reverse([1,3,5,2,4]))

print(reverse([1,9,6,4]))
```

Task 2.5

The library bought some new books and they are recorded in the file newbooks.csv. Using the function(s) you have written, write a procedure which

[3]

- uses bubble sort to sort the new books starting with the most recent publication, and
- saves the sorted new books array in a csv

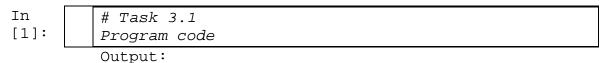
Save your csv file as

```
YEAR_<your name>_<center number>_<index number>.csv [5]
```

Save your Jupyter notebook for Task 2.

3. A programmer is writing a class, LinkedList, to represent a linked list of words. A linked list is a collection of data elements, whose order is not given by their physical placement in memory. Instead, each element points to the next.

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 3.1

The LinkedList class has two attributes:

- head is a pointer which points to the first element in the linked list,
- size is the number of elements in the linked list,

and the following methods:

- insert (word, p) inserts word into the linked list so that it is located in the p-th element in the linked list. If p is larger than the size of the linked list, insert word at the end of the linked list:
- delete(p) deletes the p-th element in the linked list. If p is larger than the size of the linked list, delete the last element in the linked list;
- search(word) returns a Boolean value: True if word is in the linked list, False if not in the linked list;
- to_String() returns a string containing a suitably formatted list with the elements separated by a comma and a space, e.g. in the form: apple, carrot, banana

Write program code for the class LinkedList. [15]

Design a test plan and write program code to fully test the insert and search methods.

[5]

Task 3.2

Write a subclass Stack using LinkedList as its superclass.

Additional push and pop methods are to be defined on the Stack class:

- push(word) will insert word to the top of the stack
- pop() will delete the top element from the stack

Write program code for the Stack class. Test your Stack class by writing program code which does the following:

- create an empty stack
- push 'apple', 'pear', 'carrot' onto the stack, in that order
- pop from the stack
- display the elements in order

[3]

Task 3.3

Write a subclass Queue using LinkedList as its superclass.

Additional enqueue and dequeue methods are to be defined on the Queue class:

- enqueue (word) will insert word to the end of the queue
- dequeue() will delete the first element in the queue

Write program code for the Queue class. Test your Queue class by writing program code which does the following:

- create an empty queue
- enqueue 'apple', 'pear', 'carrot' to the queue, in that order
- dequeue from the queue
- display the elements in order

[3]

Save your Jupyter notebook for Task 3.

4. A school stores students' health information in a database, students.db, provided with this question. Each student has a unique StudentID and can have at most one health record in the database.

There are two tables:

- Student(StudentID, Name, Gender)
- StudentHealthRecord(StudentID, Weight, Height)

The task is to use Python to create a web application.

Save your program code as

```
Task4_<your name>_<center number>_<index_number>.py
```

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

```
Task4_<your name>_<center number>_<index_number>
```

Task 4.1

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

Student health records

Health record statistics

Add health record

Save your template file as Task4_1.html

Save your program code in

```
Task4_<your name>_<center number>_<index_number>.py
```

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

```
Task4 1 <your name> <center number> <index number>.html [4]
```

Task 4.2

Write an SQL query that shows:

- all students' names, gender, weight and their height
- NULL value for weight and height for students without a health record
- sorted by gender in ascending order, then names in descending order

The results of the query should be shown on a web page in a table that:

- lists the name, gender, weight and height of each student
- has the results shown in ascending order of gender, then in descending order of names

The web page should be accessed from the menu option Student health records from Task 4.1

Save all your SQL code as

Save your template file as Task4_2.html

Save your program code in

Task4_<your name>_<center number>_<index_number>.py

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

Task 4.3

Write SQL statement(s) that shows:

- the total number of male and female students
- the average weight of male and female students
- the average height of male and female students

The results of the query should be shown on a web page in a table that shows:

- the total number of male and female students
- the average weight of male and female students
- the average height of male and female students

The web page should be accessed from the menu option Health record statistics from Task 4.1

Save all your SQL code as

Save your template file as Task4_3.html

Save your program code in

Task4_<your name>_<center number>_<index_number>.py

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

Task 4.4

Write an SQL statement that inserts a student record into

- Student table with name 'Helen' and gender 'F'
- StudentHealthRecord table with student ID 12, weight 48.7 and height 1.72

Create a web page using the template Task4_4.html provided to add a new student health record into the database, students.db. Make the necessary changes in the template file provided if required.

The web page should be accessed from the menu option Add health record from Task 4.1

Save all your SQL code as

Save your template file as Task4_4.html

Save your program code in

Task4_<your name>_<center number>_<index_number>.py

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