RIVER VALLEY HIGH SCHOOL
YEAR 6 PRELIMARY EXAMINATION

H2 COMPUTING 9597/01
Paper 1
14 AUGUST 2017

3 HOUR 15 MINUTES

NAME _____

CLASS 6 ( )

INDEX NO. _____

READ THESE INSTRUCTIONS FIRST **DO NOT OPEN THIS BOOKLET UNTIL YOU ARE TOLD TO DO SO.**		
Type in the EVIDENCE.DOC document the following: • **Candidate details** • **Programming language used** Answer **all** questions. All tasks must be done in the computer laboratory. You are not allowed to bring in or take put any pieces of work or materials n paper or electronic media or in any other form. All tasks and required evidence are numbered. The number of marks is given in brackets [ ] at the end of the task. Copy and paste required evidence of program listing and screenshots into the EVIDECE.DOC document **At the end of the examination, print out your EVIDENCE.DOC document and fasten your printed copy securely together. Save all your source file in the thumb drive given.**		**FOR EXAMINERS' USE** **1** **/20** **2** **/20** **3** **/10** **4** **/20** **5** **/20** **TOTAL** **/90**

This Question Paper consists of **14** printed pages.

1. Results management

For this task, you are required to read the text file ***"overall_grades.txt"*** which contains the subject grades of the first and second semester examination of 50 students.

The format of the text file is as follow:

```
<name>;<subject1>:<grade1>,<grade2>;<subject2>:<grade1>,<grade2>;<subject3>:<grade1>,<grade2>,…etc;\n
```

For example:

```
Rufus Schuck;EL:C,B;CL:C,C;Math:C,E;Bio:B,F;
```

Student Rufus Schuck takes 4 subjects. They are English (EL), Chinese (CL), Math (Math) and Biology (Bio). His 1st and 2nd semester grades for English is C and B respectively.

Task 1.1

Write the program code for the function `process_data()` which reads the text file ***"overall_grades.txt"*** and returns a dictionary which has the following format.

```
{'Rufus Schuck': {'EL': ['C', 'B'], 'CL': ['C', 'C'], 'Math': ['C', 'E'], 'Bio': ['B', 'F']}, ...}
```

The returned dictionary uses the name of student as key and another dictionary as its return value. This "another" dictionary then uses the subjects and a list containing the grades of the 1st and 2nd semesters as its key and value respectively.

Evidence 1

The program code for the function `process_data()`. [4]

Task 1.2

Write a function `have_n_improvement(n)` which takes in an integer `n` and returns a list of student names who have improvement in exactly `n` subject(s). Using the example in Task 1.1, `'Rufus Schuck'` shows improvement in only English. So, if `n` is 1, `'Rufus Schuck'` must be included in the student name list.

Evidence 2

The program code for the function `have_n_improvement(n)`. [3]

Task 1.3

Write a function `have_improvement_in_all_subjs()` which returns a list of student names who have improvement in all his/her subjects.

Evidence 3

The program code for the function `have_improvement_in_all_subjs(n)`. [3]

Task 1.4
Write a function `count_combi(combi)` which takes a list of subjects `combi` as input and returns the total number of students who take the subject combination in `combi`. Take note that students who take more subjects than what is indicated in `combi` are included.

For example:
```
>>> count_combi(['EL', 'MATH'])
>>> 50
```

Evidence 4
The program code for the function `count_combi(n)`. [4]

Task 1.5
Using your solution in Task 1.4, find the total number of students who take "Chem" and "Bio" but not "Physics".

Evidence 5
The program code to find what is required in Task 1.5. [2]

Task 1.6
Write a function `calculate_GPA(name)` which takes a string `name` as input and calculates the GPA of the student indicated by `name`.
The points for each grade are as follow:
`{'A':5, 'B':4, 'C':3, 'D':2, 'E':1, 'F':0}`
The 1st and 2nd semester grades in the data has the same weightages.

To calculate GPA,

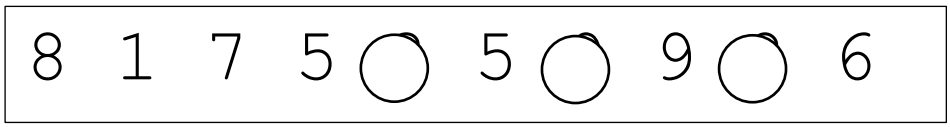
- Find the average points achieved of each subject.
  - e.g. if a student gets C and D in "CL", his average points for "CL" is 3 + 2 = 2.5
- Add up the average points of each subject
- Divide the total by the number subjects the student takes
  - e.g. `'Claudette Bode': {'EL': ['C', 'D'], 'CL': ['D', 'A'], 'Math': ['A', 'D'], 'Phy': ['A', 'D'], 'Bio': ['D', 'C']},`
  - [(3 + 2)/2 + (2 + 5)/2 + (5 + 2)/2 + (2 + 5)/2 + (2 + 3)/2]/5 = 3.1
- GPA for `'Claudette Bode'` is 3.1

Evidence 6
The program code for the function `calculate_GPA(name)`. [4]

2. Missing phone number digits

Your friend (Mary) made an oversea friend (Jenny) yesterday while she was studying in the regional library. To remain in contact, Jenny gave Marry her phone number and told Mary that it is also a valid 10 digits International Standard Book Number (ISBN).

After Jenny had left, Mary accidentally punched a few holes on the phone number while she was arranging her notes using the library's hole puncher machine. It was too difficult for Mary to recover the lost pieces of paper. So, Mary decided to look for you to recover the missing digits instead. Below is the picture of the paper.



To recover the missing digits, you need to understand how ISBN-10 works. The steps to check for a valid ISBN-10 is as follow:

1. Each of the digits of the ISBN-10 is first multiplied by a number in a sequence from 10 to 1 (Take note that the last digit of the ISBN-10 can be '**X**' which holds a value of 10).
2. All the 10 digits are then added up together.
3. If the sum is divisible by 11, it is a valid ISBN-10, else it is invalid.

For example, to check if '997150233X' is a valid ISBM-10:
Step 1 and 2
*(9x10) + (9x9) + (7x8) + (1x7) + (5x6) + (0x5) + (2x4) + (3x3) + (3x2) + (**X**x1)*
*= (9x10) + (9x9) + (7x8) + (1x7) + (5x6) + (0x5) + (2x4) + (3x3) + (3x2) + (**10**x1)*
= 297
Step 3
297 is a multiple of 11. Therefore, '997150233X' is a valid ISBN-10.

Task 2.1

Write a function `is_valid_ISBN10(isbn10)` which takes a string `isbn10` as input and returns `True` if `isdn10` is a valid ISBN-10 and `False` otherwise. You are to validate the input string `isbn10` and print to terminal the issue related to the `isbn10` string. The possible issues to be printed to terminal includes:

- `ISBN-10 must only have 10 digits.`
- `ISBN-10 can only contain numbers and the character 'X'.`
- `'X' can only be found at the last digit of isbn10.`
- `Wrong check digit.`
- `Valid ISBN-10.`

Evidence 7

The program code for the function `is_valid_ISBN10(isbn10)`. [6]

Task 2.2

Write a function `possible_ISBN(isbn)` which takes a query string `isbn10` as input and returns a list of possible ISBN-10. The query string `isbn10` has 3 missing digits replaced with a `'?'` and they can be at any positions of the query string.

For example,

```
>>> possible_ISBN("9971??23?X")
>>> ['997100237X', '997102232X', '997103235X', '997104238X',
'997105230X', '997106233X', '997107236X', '997108239X',
'997109231X', '997110234X', '997111237X', '997113232X',
'997114235X', '997115238X', '997116230X', '997117233X',
'997118236X', '997119239X', '997120231X', '997121234X',
'997122237X', '997124232X', '997125235X', '997126238X',
'997127230X', '997128233X', '997129236X', '997130239X',
'997131231X', '997132234X', '997133237X', '997135232X',
'997136235X', '997137238X', '997138230X', '997139233X',
'997140236X', '997141239X', '997142231X', '997143234X',
'997144237X', '997146232X', '997147235X', '997148238X',
'997149230X', '997150233X', '997151236X', '997152239X',
'997153231X', '997154234X', '997155237X', '997157232X',
'997158235X', '997159238X', '997160230X', '997161233X',
'997162236X', '997163239X', '997164231X', '997165234X',
'997166237X', '997168232X', '997169235X', '997170238X',
'997171230X', '997172233X', '997173236X', '997174239X',
'997175231X', '997176234X', '997177237X', '997179232X',
'997180235X', '997181238X', '997182230X', '997183233X',
'997184236X', '997185239X', '997186231X', '997187234X',
'997188237X', '997190232X', '997191235X', '997192238X',
'997193230X', '997194233X', '997195236X', '997196239X',
'997197231X', '997198234X', '997199237X']
```

Evidence 8

The program code for the function `possible_ISBN(isbn)`. [6]

Task 2.3

Examine the missing numbers in the paper closely and eliminate impossible digit choices. For example, the three missing digits cannot be ‘1’ as the three missing digits in the paper have round edges at the top. Modify your solution in Task 2.2 and suggest these possible ISBN-10 to Mary.

Evidence 9

The screenshot of the suggested possible ISBN-10 that you will give to Mary. [2]

Task 2.4
Write a menu to help other to generate possible ISBN-10. The menu should have the following functions.

```
Menu
A) Input a ISBN-10 query string
B) Display all possible ISBN-10
C) Quit
```

You should validate user inputs and call for re-entry when an invalid option is given.

Evidence 10
Program code for the procedure `menu()`. [6]

3. Unsorted Linked List

The classes `Node` and `LinkedList` is partially implemented for you according to the specification given below. The attribute `start` of class `LinkedList` stores the first node instance of the linked list. If the linked list has no node, `start` has the value of `None`. Functions in bold are not implemented.

Node	**LinkedList**
-data : STRING -next : Node	-start : Node
+constructor() +set_data(data) +get_data() +set_next(node) +get_next() +__str__()	+constructor() +set_start(node) +get_start() **+insert_data(data)** **+transfer(linkedlist)** **+delete_pos(pos)** +display()

Task 3

Implement the following `LinkedList` class functions according to its specification given:

- `+insert_data(node)`
- `+transfer(llist)`
- `+delete_pos(pos)`

Function	**Specification**
`insert_data(data)`	This function takes a string `data` as input and uses its value to create a new `Node` instance. The new `Node` instance is then inserted as the first node of the linked list.

`transfer(llist)`	This function takes a `LinkedList` instance `llist` as input and transfers all its `Node` instances in the same order to the `LinkedList` instance that calls for this function.
	For example:
	`A:'boy'->'man'->'woman'`
	`B:'girl'->'baby'`
	`>>> A.transfer(B)`
	`A: 'boy'->'man'->'woman'->'girl'->'baby'`
	`B: None`

`delete_pos(pos)`	This function takes in an integer `pos` as input and deletes the node at position `pos`. Take note that the first node of the linked list is position 1.
	For example:
	`A:'boy'->'man'->'woman'`
	`>>> A. delete_pos(2)`
	`A:'boy'->'woman'`

Evidence 11
Program code for:

- `insert_data(node)` [3]
- `transfer(linkedlist)` [3]
- `delete_pos()` [4]

4. Sorting Algorithms

The pseudocode procedure below is given a list of integers in random order, the procedure then outputs a sorted list in ascending order.

```
1.  FUNCTION Merge_Sort (ARRAY: arr)
2.      n = SIZE(arr)
3.      IF n < 2:
4.          RETURN arr
5.      END IF
6.      left = Merge_Sort(PARTITION(arr, 0, n DIV 2))
7.      right = Merge_Sort(PARTITION(arr, n DIV 2, n))
8.      RETURN Merge(left, right)
9.  END FUNCTION
10.
11. FUNCTION Merge (ARRAY: left, LIST: right)
12.     DECLARE results: ARRAY[SIZE(left) + SIZE(right)]
13.     DECLARE i, j, index : INTEGER
14.     i, j, index = 0, 0, 0
15.     WHILE i < SIZE(left) and j < SIZE(right)
16.         IF left[i] < right[j]:
17.             results[index] = left[i]
18.             i = i + 1
19.         ELSE:
20.             results[index] = right[j]
21.             j = j + 1
22.         END IF
23.         index = index + 1
24.     END WHILE
25.     WHILE i < SIZE(left)
26.         results[index] = left[i]
27.         i = i + 1
28.         index = index + 1
29.     END WHILE
30.     WHILE j < SIZE(right)
31.         results[index] = right[j]
32.         j = j + 1
33.         index = index + 1
34.     END WHILE
35.     RETURN results
36. END FUNCTION
```

Task 4.1
Write program code to implement the given procedure.
Execute the function using the following list as the parameter.
`[1, 15, 2, 4, 3, 9, 7, 10]`

Evidence 12
Program Code.
Screenshot showing the output. [6]

Evidence 13
Analyze and state the time complexity of the algorithm in big O notation. [1]

Task 4.2
Bubble sort is a common sorting algorithm, below is an implementation of in-place bubble sort using recursion. There are three missing lines in the pseudocode, indicated as `A`, `B` and `C`.

```
1.  PROCEDURE Bubble_Sort (ARRAY: arr, INTEGER: n)
2.      IF ______A______:
3.          RETURN
4.      END IF
5.
6.      FOR i FROM 0 TO (n-1):
7.          IF ______B______:
8.              temp = arr[i]
9.              arr[i] = arr[i+1]
10.             arr[i+1] = temp
11.         END IF
12.     END FOR
13.
14.     ______C_______
15. END PROCEDURE
```

Execute the procedure to sort the following list.
`[1, 15, 2, 4, 3, 9, 7, 10]`

Evidence 14
Complete the missing lines `A`, `B` and `C`. [3]

Insert a counting mechanism to count the number of comparisons made, and return the value as an integer. Implement the new recursive bubble sort.

Evidence 15
Program Code.
Screenshot showing the output of the sorted list and count value. [4]

Evidence 16
Analyze and state the time complexity of the algorithm in big O notation. [1]

Task 4.3
Due to the nature of bubble sort, if no swaps are observed in a given iteration, then there is no need for the next iteration. Implement the improved bubble sort. Execute the function to sort the following list.

```
[1, 15, 2, 4, 3, 9, 7, 10]
```

Evidence 17
Program Code.
Screenshot showing the output of the sorted list and count value. [3]

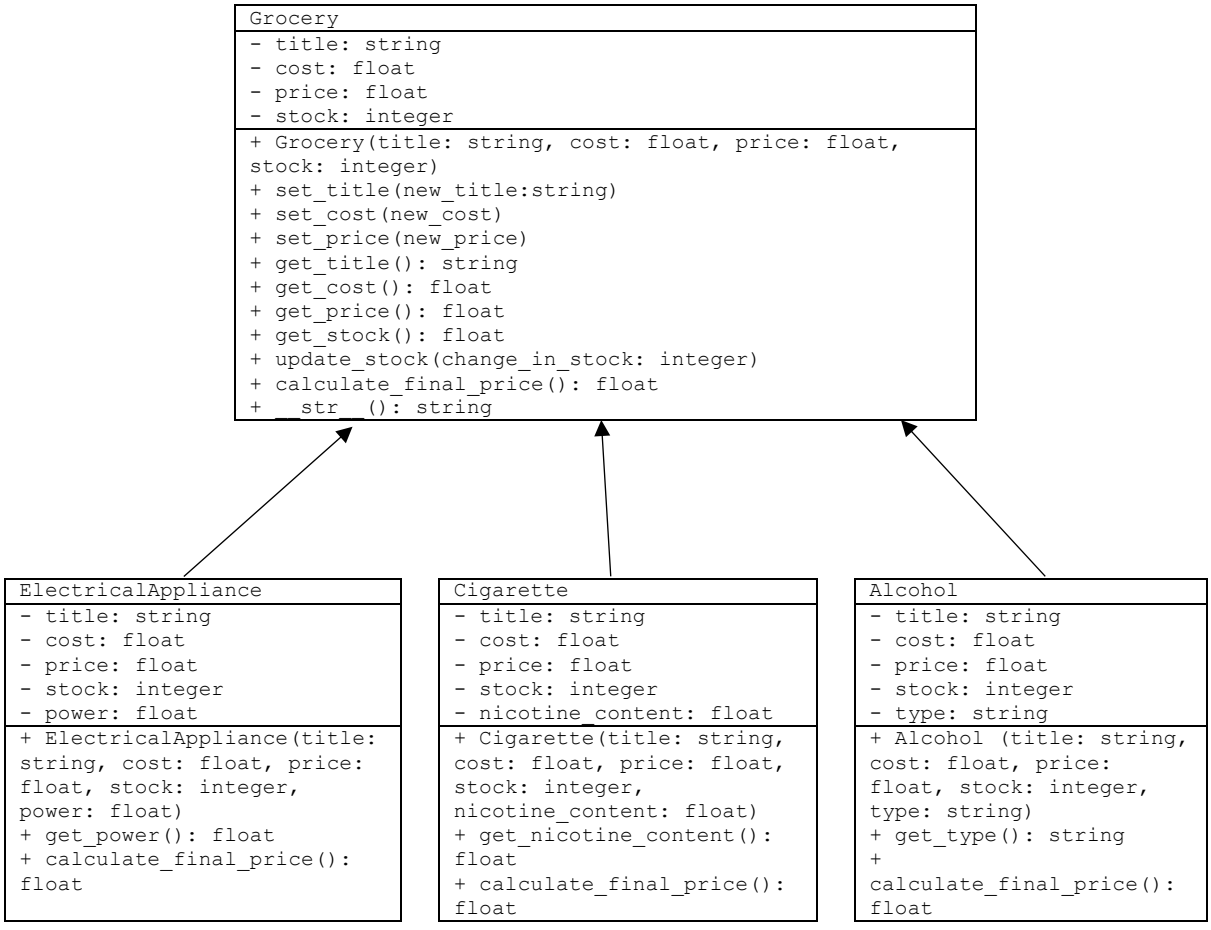
5. Grocery Store Manager

The grocery shop in the neighborhood ask your help to create an application to manage the grocery store.

First, you are tasked to work on an object-oriented solution to store all the grocery details. The `title` of the grocery item, `cost`, `price` and `stock` of each grocery is recorded. Besides the normal groceries, the shop identifies three unique types of grocery too, namely:

- Electrical Appliance: there is a need to indicate the `power` of the product to understand its energy consumption rate.
- Cigarette: it is important to track the `nicotine content` of various kinds of cigarette.
- Alcohol: there are distinct `types` of alcohols such as wine or beer.

Below is an UML class diagram for your reference.



Task 5.1
Implement the classes of `Grocery`, `ElectricalAppliance`, `Cigarette` and `Alcohol` with object-oriented programming.

Evidence 18
Program Code for four classes. [7]

Task 5.2
In country S, purchasing all groceries will incur a 7% Goods and Services Tax(GST).

To promote healthy living, additional tax have been imposed to cigarettes and alcohols:

- Cigarette: additional 60% tax
- Wines: additional 50% tax
- Beers: additional 20% tax

For example: the price of one packet of "Yun Yan" cigarette is $23.00, the final price can be calculated by: $23.00 x 160% x 107% = $39.38

In addition, to support the "save energy movement", all electrical appliance with a power less than or equals to 10Watt was set to be sold at 80% of its original price.

Implement the function `calculate_final_price()` which includes the above mentioned tax and promotion into consideration.

Implement the `__str__()` function which returns a string in the following format (You may refer to the `test_function_5_1()` to understand the formatting):

```
Title                |    Cost |   Price |  Stock |   Final Price
```

For example, Yun Yan's cost is $16.50, price is set at $23.00, the current stock is 4 and final price is $39.38. The __str__() function should return the following string:

```
Yun Yan              | $ 16.50 | $ 23.00 |      4 | $       39.38
```

Evidence 19
Program Code for `calculate_final_price()` and `__str__()`.
Screenshot showing output of `test_function_5_1()`. [6]

Task 5.3

Implement a class `StoreManager` which keep track of a list of grocery items, `curr_item_list`. The `StoreManager` should have the following class functions:

Function	Description
`sell_item(sold_item)`	`sold_item` is a tuple containing 2 elements: the `title` of the item and the `quantity` sold. You may assume the title of item is always valid and the quantity sold is always smaller than the current stock. The function should decrease the current stock of the sold items. Upon completion, it should print out a string containing the following information: `Title | Unit Price | Quantity Sold | Subtotal` The function should return a float containing the `sub_total` value.
`sell_items(sold_item_list)`	`sold_item_list` is a list of tuples; each tuple containing the item `title` and `quantity` sold. The function should print out a table displaying information for all sold items in the following format: `Title | Unit Price | Quantity Sold | Subtotal` The summary should end with a line indicating the overall total value of items sold in this transaction.
`stock_check()`	When this function is called, it should check the list of all grocery items and print out a summary of items with current stock value below 5. This indicates the need for stocking up these items soon. A summary table should be printed in the following format: `Title | Unit Cost | Quantity Left`

Evidence 20

Program Code for class functions of `StoreManager` class:

- `sell_item` [3]
- `sell_items` [2]
- `stock_check` [2]

Screenshot showing output of `test_function_5_2()`. [2]