RIVER VALLEY HIGH SCHOOL
JC2 PRELIM EXAMINATION

H2 COMPUTING 9597
Paper 1
21 AUGUST 2019

3 HOUR 15 MINUTES

NAME _____

CLASS J18 ( )

INDEX NO. _____

READ THESE INSTRUCTIONS FIRST

DO NOT OPEN THIS BOOKLET UNTIL YOU ARE TOLD TO DO SO.

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 or paper or electronic media or in any other form.

All tasks are numbered.

The number of marks is given in brackets [ ] at the end of the task.

Approved calculators are allowed.

At the end of the examination, print out your evidence file and save all your source files in the thumb drive provided.

FOR EXAMINERS' USE	
1	/25
2	/15
3	/10
4	/40
TOTAL	/90

This Question Paper consists of **16** printed pages.

1. CID3 Team Grouping

In this question, you will help the CID3 students in forming CID3 groups for their projects.

In ***"student_cid.txt"***
There are three fields on each line which indicates `name`, `role` and `gender` of 50 cid3 students. The fields are separated by '`;`'

```
Rufus Schuck;Coder;F
Ione Wolfe;Dealer;F
Hillary Curl;Coder;M
…
```

Task 1.1
Implement the function `read_data(filename)` which takes `filename` as a string and returns a 2-dimension list that follows the format as shown in the example below.

```
>>> read_data("student_cid.txt")
[['Rufus Schuck', 'Coder', 'F'],
 ['Ione Wolfe', 'Dealer', 'F'],
 ['Hillary Curl', 'Coder', 'M'],…]
```

Evidence 1
Program code of function `read_data`. [2]

Task 1.2
Implement the function `gender_count(cid_student_lst, is_female)` which takes a list `cid_student_lst` and a boolean `is_female` as inputs and returns the number of female students in `cid_student_lst` if `is_female` is `True`, otherwise return the number of male students. `cid_student_lst` is the list obtained in **Task 1.1**.

Evidence 2
Program code of function `gender_count`. [2]

Evidence 3
Screenshot of the output of the following:

```
cid_student_lst = read_data("student_cid.txt")
print(gender_count(cid_student_lst, True))
print(gender_count(cid_student_lst, False))
```

[1]

Task 1.3

Implement the procedure `role_statistics(cid_student_lst)` which takes a list `cid_student_lst` as input and output the number of students for each role in the following format. (There are more than 5 types of roles.)

For example:

```
>>> cid_student_lst = read_data("student_cid.txt")
>>> role_statistics(cid_student_lst)
Role           Number
Coder          11
Dealer         13
Designer       14
Empathizer     14
Maker          12
```

Take note that the roles and numbers shown above is just an example.

Evidence 4

Program code of procedure `role_statistics`. [3]

Evidence 5

Screenshot of the output of the following:

```
cid_student_lst = read_data("student_cid.txt")
role_statistics(cid_student_lst)
```

[1]

Task 1.4

Implement the function `form_random_group(cid_student_lst)` which takes a list `cid_student_lst` as input and returns a list consists of 5 student names. This list of students forms a group and must consist of one coder, one maker, one dealer, one empathizer and one designer. The student picked for each role must be random. If there is not sufficient roles or students to form a group, return an empty list.

For example:

```
>>> cid_student_lst = read_data("student_cid.txt")
>>> form_random_group(cid_student_lst)
['Fredricka Gormley', 'Jalisa Stoudemire', 'Laverna Halpern',
'Chadwick Griffin', 'Abdul Boland']
```

Note:

```
Fredricka Gormley is a coder
Jalisa Stoudemire is a dealer
Laverna Halpern is a designer
Chadwick Griffin is an empathizer
Abdul Boland is a maker
```

Evidence 6

Program code of procedure `form_random_group`. [5]

Evidence 7
Screenshot of the output of the following:

```
cid_student_lst = read_data("student_cid.txt")
for i in range(3):
    print(form_random_group(cid_student_lst))
```

[1]

Task 1.5
Implement the function `remove_students` which takes `cid_studnet_lst` and `one_cid_group` as inputs where `cid_studnet_lst` is the list obtained from **task 1.1** and `one_cid_group` is the list of 5 student names obtained from **task 1.4**. The function removes 5 records in `cid_studnet_lst` specified by the student names in `one_cid_group` and returns the removed records in a list.

For example:

```
>>> cid_student_lst = read_data("student_cid.txt")
>>> one_cid_group  = form_random_group(cid_student_lst)
>>> one_cid_group
['Rufus Schuck', 'Kathlene Collar', 'Luanne Lett', 'Phyliss Rolen', 'Tobias Kimmer']
>>> remove_students(cid_student_lst, one_cid_group)
[['Rufus Schuck', 'Coder', 'F'], ['Kathlene Collar', 'Empathizer', 'M'], ['Luanne Lett', 'Dealer', 'F'], ['Phyliss Rolen', 'Maker', 'M'], ['Tobias Kimmer', 'Designer', 'F']
>>> len(cid_student_lst)
45
```

After `remove_students(cid_student_lst, one_cid_group)` is executed `cid_student_lst` should not contain any records with students name `Fredricka Gormley`, `Jalisa Stoudemire`, `Laverna Halpern`, `Chadwick Griffin` and `Abdul Boland`. Since the 5 names are removed. `cid_student_lst` should now have 45 records.

Evidence 8
Program code of function `remove_students`. [4]

Evidence 9
Screenshot of the output of the following code.

```
def test_15():
    print("------Task 1.5------")
    cid_student_lst = read_data("student_cid.txt")
    one_cid_group = form_random_group(cid_student_lst)
    removed_records = remove_students(cid_student_lst, one_cid_group)
    print("removed records")
    for removed_record in removed_records:
        print(removed_record)
    print("remaining records")
    for cid_student in cid_student_lst:
        print(cid_student)

test_15()
```

[1]

Task 1.6

Using your solutions in **task 1.1**, **1.4** and **1.5**, write a procedure `form_max_cid_group` which takes a list `cid_student_lst` as input and write to a file named "`result.txt`" the suggested maximum number of CID3 groups that can be formed from `cid_student_lst`. The content in ***"result.txt"*** should also include the group number and its group members. For example, the content of ***"result.txt"*** can be:

```
Group 0
Rufus Schuck Coder F
Lashawna Meals Dealer M
Phyliss Rolen Maker M
Laverna Halpern Designer F
Apryl Soileau Empathizer F
Group 1
Claudette Bode Maker F
Angle Linck Coder F
Grazyna Kitzman Designer M
Virgilio Britt Dealer F
Dannette Raasch Empathizer F
Group 2
Carolann Kintner Designer M
Ola Markell Empathizer F
Jaye Galle Maker F
Lanita Sciortino Coder M
Joella Wessner Dealer F
Group 3
Hertha Dossantos Dealer F
Chadwick Griffin Empathizer M
Fredricka Gormley Coder F
Marcella Daigneault Designer F
Farah Quon Maker F
Group 4
Hillary Curl Coder M
Elvia Dubrey Designer F
Terrence Shannon Empathizer M
Luanne Lett Dealer F
See Borne Maker F
Group 5
Toney Mcnab Coder M
Jalisa Stoudemire Dealer M
Abdul Boland Maker M
Russell Gillison Designer F
Reiko Stack Empathizer F
```

Evidence 10

Program code of procedure `form_max_cid_group`. [4]

Evidence 11

Screenshot of the content of *"result.txt"*. [1]

2. EAN-13

EAN-13 (European Article Number) barcode a standard describing a barcode symbology and numbering system used in global trade to identify a specific retail product type, in a specific packaging configuration, from a specific manufacturer.

EAN check digits are calculated by summing each of the odd position numbers multiplied by 3 and then by adding the sum of the even position numbers.

Numbers are examined going from right to left, so the first odd position is the last digit in the code. The final digit of the result is subtracted from 10 to calculate the check digit.

For example,

```
EAN(first 12 digits) =  400638133393
Even digits          =  4 + 0 + 3 + 1 + 3 + 9      = 20
Odd digits x 3       = (0 + 6 + 8 + 3 + 3 + 3) x 3 = 69
Total                = 20 + 69                     = 89
Check digit          = 10 - 9                      = 1
```

Therefore, the valid EAN number is `4006381333931`.

Task 2.1

Implement the ***iterative*** function `EAN` that takes in a string `ean12` which is the first 12 characters of a valid EAN number and returns the full valid EAN with the check digit in string.

For example:

```
>>> EAN("400638133393")
"4006381333931"
>>> EAN("590123412345")
"5901234123457"
```

Evidence 12

Program code for `EAN`. [3]

Evidence 13

Screenshot of the output of the following code. [1]

```
print(EAN("400638133393"))
print(EAN("590123412345"))
print(EAN("950110153000"))
print(EAN("007567816412"))
print(EAN("123456789123"))
print(EAN("563643712973"))
```

Task 2.2
Implement the function `EAN_rec` which is the ***recursive*** version of task 2.1 **.**

Evidence 14
Program code for `EAN_rec`. [3]

Task 2.3
Implement the function `generate_n_random_EAN(n)` that takes an integer `n` as input and return a list that contains `n` random valid EAN numbers in string.

For example:
```
>>> generate_n_random_EAN(5)
['9399783016850',       '7126497037138',       '7859230985143',
'4663965860605', '0075678464126']
```

Evidence 15
Program code for `generate_n_random_EAN`. [3]

Evidence 16
Screenshot of the output of the following code: [1]
```
print(generate_n_random_EAN(5))
```

Task 2.4
Implement the function `quick_sort_10_EAN()` that performs quicksort on 10 randomly generated valid EANs and returns the sorted list of EANs in string.

For example:
```
>>> quick_sort_10_EAN()
['0777557883249',       '1830930669218',       '1932904647625',
'3257925382651',       '6272017045297',       '6715598129708',
'7248472619815',       '7660010013945',       '9810395262430',
'9870932286909']
```

Evidence 17
Program code for `quick_sort_10_EAN`. [3]

Evidence 18
Screenshot of the output of the following code: [1]
```
print(quick_sort_10_EAN())
```

3. Minimum Heap

A minimal heap is a binary tree that always maintains the smallest data item at its root node. In this question, the `class minHeap` is implemented using a 1D array with each child node index calculated using the following formula.

```
left_child_ptr = node_ptr x 2 + 1
right_child_ptr = node_ptr x 2 + 2
```

class minHeap attributes	**Description**
count (INTEGER)	It stores the number of data item currently in minheap.
size (INTEGER)	It stores the maximum number of data item minHeap can take.
tree (ARRAY OF INTEGER)	It is a 1D array that stores the data items as nodes in minHeap. If a data item doesn't exist, it is represented by -1.

Task 3.1
The pseudo-code of the class procedure `add(newItem)` is given in file ***"task31.txt"***. Use it to implement the class procedure `add`.

```
PROCEDURE add(newItem)
    IF minHeap is not full THEN

        tree[count] <- newItem
        curr_ptr <- count
        parent_ptr <- QUOTIENT((curr_ptr - 1) DIV 2)

        REPEAT
            SWAP (tree[parent_ptr], tree[curr_ptr])
            curr_ptr <- parent_ptr
            parent_ptr <- QUOTIENT((curr_ptr - 1) DIV 2)
        UNTIL curr_ptr EQUAL TO 0 OR tree[parent_ptr]<= newItem

        INCREMENT count BY 1
    ELSE
        OUTPUT "Heap is full. Cannot add."
    END IF
END PROCEDURE
```

Evidence 19
Program code for `add`. [3]

Task 3.2

The class function `remove_minimum` is implemented for you in file ***"T3.py"***. This function removes the data item at the root node of the minimum heap and returns the data item.

Your task is to implement the class function `sort` which returns a list consists of all the data items stored in the minimum heap in increasing order. Take note that after `sort` is executed, the minimum heap becomes empty.

For example:

```
def test():
    test_value = [58, 36, 3, 9, 87]
    h1 = minHeap(5)
    for value in test_value:
        h1.add(value)
    print(h1.sort())

>>> test ()
[3, 9, 36, 58, 87]
```

Evidence 20

Program code for the class function `sort`. [2]

Evidence 21

Screenshot of the output of the following code: [1]

```
def test_32(n):
    test_value = random.sample(range(1,100), n)
    h1 = minHeap(n)
    for value in test_value:
        h1.add(value)
    print(h1.sort())

print("task 3.2")
print("1st run")
test_32(15)
print("2nd run")
test_32(15)
print("3rd run")
test_32(15)
```

Task 3.3

Implement the class procedure `display_all_paths` which displays all paths from the root of minimum heap to all its leaves. *Hint: The minimum heap in this question is implemented using a complete binary tree. This means that the `tree` array indices from `0` to `count-1` contain all the data items of the minimum heap.*

For example:

```
def test_33(n):
    test_value = random.sample(range(1,100), n)
    h1 = minHeap(n)
    for value in test_value:
        h1.add(value)
    h1.display_all_paths()

>>> test_33(10)
5 16 21 40
5 16 21 29
5 16 34 94
5 49 96
5 49 69
```

Evidence 22

Program code for `display_all_paths`. [3]

Evidence 23

Screenshot of the output of the following code: [1]

```
print("task 3.3")
print("1st run")
test_33(5)
print("2nd run")
test_33(10)
print("3rd run")
test_33(15)
```

4. Begemed

Begemed is a casual game which is played on ruled grids. The player is required to swap a gem in one of the four possible directions, namely “up”, “down”, “left”, “right”; after the swap, if a row or a column of 3 or more gems are formed, it’s considered a valid move and the connected gems will be destroyed. Otherwise, it’s considered an invalid move. Note that diagonal directions are not counted.

Below are some examples of the game demonstrated in a 5x5 grid. The letters "d", "s", "t", "r", "e" represents Diamond, Sapphire, Topaz, Ruby, Emerald respectively.

	0	1	2	3	4
0	r	r	d	e	d
1	s	d	r	**d**	**t**
2	d	d	r	s	d
3	e	r	t	r	r
4	t	e	e	s	d

1. A swap between "d" at (1, 3) and "t" at (1, 4) is considered a valid move.

	0	1	2	3	4
0	r	r	d	e	**d**
1	s	d	r	**t**	**d**
2	d	d	r	s	**d**
3	e	r	t	r	r
4	t	e	e	s	d

2. The grid will look like this after the swap. There are three "d"s found at (0, 4), (1, 4) and (2, 4) forming a column and they will be destroyed.

	0	1	2	3	4
0	r	r	d	e	_
1	s	d	r	**t**	_
2	d	d	r	s	_
3	e	r	t	r	r
4	t	e	e	s	d

3. After the gems are destroyed, the respective grid cells will be filled with "_" temporarily.

Some other valid swaps are:

	0	1	2	3	4
0	r	**r**	**d**	e	d
1	s	d	r	d	t
2	d	d	r	s	d
3	e	r	t	r	r
4	t	e	e	s	d

	0	1	2	3	4
0	r	**d**	**r**	e	d
1	s	d	r	t	d
2	d	d	r	s	d
3	e	r	t	r	r
4	t	e	e	s	d

	0	1	2	3	4
0	r	_	_	e	d
1	s	_	_	t	d
2	d	_	_	s	d
3	e	r	t	r	r
4	t	e	e	s	d

"r" at (0, 1) and "d" at (0, 2) can be swapped.

	0	1	2	3	4
0	r	r	d	e	d
1	s	d	r	d	t
2	d	d	**r**	s	d
3	e	r	**t**	r	r
4	t	e	e	s	d

	0	1	2	3	4
0	r	r	d	e	d
1	s	d	r	d	t
2	d	d	**t**	s	d
3	e	r	**r**	r	r
4	t	e	e	s	d

	0	1	2	3	4
0	r	r	d	e	d
1	s	d	r	d	t
2	d	d	**t**	s	d
3	e	_	_	_	_
4	t	e	e	s	d

"r" at (2, 2) and "t" at (3, 2) can be swapped.

	0	1	2	3	4
0	r	r	d	e	d
1	s	d	r	d	t
2	d	d	r	s	d
3	e	**r**	**t**	r	r
4	t	e	e	s	d

	0	1	2	3	4
0	r	r	d	e	d
1	s	d	r	d	t
2	d	d	r	s	d
3	e	**t**	**r**	r	r
4	t	e	e	s	d

	0	1	2	3	4
0	r	r	d	e	d
1	s	d	_	d	t
2	d	d	_	s	d
3	e	**t**	_	_	_
4	t	e	e	s	d

"r" at (3, 1) and "t" at (3, 2) can be swapped.

You are tasked to create a text-based interactive “Begemed” game in the following tasks.

Task 4.1

Implement `Begemed` class according to the UML class diagram and attributes/methods specifications given.

```
Board
- board: list
+ Board(size: int)
+ new_game(board: list)
+ check_connection(row: int, col: int): boolean
+ find_valid_moves(row: int, col: int): list
+ display(hint: boolean=False)
```

Attribute	**Specification**
`board: list`	`board` is a 2-dimensional `list` hosting the gems inside each grid.

Methods	**Specification**
`Board(n: int)`	`n` is the size used to define the dimension of the `board`. The `board` should be initialized to a (`n x n`) 2-dimensional list filled with string “_”. [2]

`new_game(new_board : list)`	`new_game` takes in a `list` named `new_board` and assign it to the class attribute `board`. The following list is provided in the python template file.
	
	`test_board = [`
	`['r', 'r', 'd', 'e', 'd'],`
	`['s', 'd', 'r', 'd', 't'],`
	`['d', 'd', 'r', 's', 'd'],`
	`['e', 'r', 't', 'r', 'r'],`
	`['t', 'e', 'e', 's', 'd']]`
	
	This method is just a temporary solution which help you in initial coding and debugging. In a later task, there will be further instructions to update its implementation. [1]

`check_connection( row: int, col: int): boolean`	`check_connection` takes in the `row` and `col` value of a particular gem, then check if there is a connection of 3 or more gems of the same type in its horizontal or vertical direction. Return `True` if such a connection is found, and `False` otherwise. [8]

Evidence 24

Program code of class `Board` and class methods up to `check_conection`. [11]

Methods	**Specification**

`find_valid_moves(row: int, col: int): list`	`find_valid_moves` takes in the `row` and `col` value of a particular gem, then attempt to swap in the four directions (up, down, left, right). If there is a new connection of 3 or more gems of the same type formed, record as a valid movement.
	
	Return a `list` containing all valid movements. An empty `list` is to be returned if no valid movement is found.
	
	For example:
	
	`  0  1  2  3  4`
	`+---+---+---+---+---+`
	`0 | r | r | d | e | d |`
	`+---+---+---+---+---+`
	`1 | s | d | r | d | t |`
	`+---+---+---+---+---+`
	`2 | d | d | r | s | d |`
	`+---+---+---+---+---+`
	`3 | e | r | t | r | r |`
	`+---+---+---+---+---+`
	`4 | t | e | e | s | d |`
	`+---+---+---+---+---+`
	
	If `find_valid_moves(0, 2)` is called, the function should return `['d', 'l']` because when down swap or left swap is performed on gem at `(0, 2)`, a new connection of 3 or more gems of the same type will be formed.
	
	[5]

`display(hint: Boolean=False)`	`display` will print out the `board` according to the sample format given. Take note that the `size` of the board can be changed and hence the grid outline should be dynamically adjusted according to its `size`.
	
	For example:
	
	`  0  1  2  3  4`
	`+---+---+---+---+---+`
	`0 | r | r | d | e | d |`
	`+---+---+---+---+---+`
	`1 | s | d | r | d | t |`
	`+---+---+---+---+---+`
	`2 | d | d | r | s | d |`
	`+---+---+---+---+---+`
	`3 | e | r | t | r | r |`
	`+---+---+---+---+---+`
	`4 | t | e | e | s | d |`
	`+---+---+---+---+---+`
	
	[7]

	`hint` is an optional argument with a default value of `False`. If `hint` is set to be `True`, the gems with valid moves should be highlighted by using the **uppercase** letters, and the valid moves for the **coordinates** and **directions** should be displayed too. For example: `(0, 1) ['r']` `(0, 2) ['d', 'l']` `(1, 2) ['u']` `(1, 3) ['u', 'r']` `(2, 2) ['d']` `(3, 0) ['d']` `(3, 1) ['r']` `(3, 3) ['l']` `  0  1  2  3  4` `+---+---+---+---+---+` `0	r	R	D	e	d |` `+---+---+---+---+---+` `1	s	d	R	D	t |` `+---+---+---+---+---+` `2	d	d	R	s	d |` `+---+---+---+---+---+` `3	E	R	t	R	r |` `+---+---+---+---+---+` `4	t	e	e	s	d |` `+---+---+---+---+---+` [2]

Evidence 25
Program code of class method `find_valid_moves` and `display`. [14]

Task 4.2
Write a texted based menu which has the following options. Validation of the user input is needed.

```
Choose an option below:
1) Validate Move
2) Toggle Hint Mode
3) Move the Gem!
4) New Game
5) Exit
```

The descriptions for the options can be found below.

Option	**Descriptions**
`Validate Move`	Ask user to input a set of `row`, `col` and `direction`. Check and feedback if this swap is valid.
`Toggle Hint Mode`	For every new game, the `hint` mode by default should be `off`. Use this option to toggle the `on` and `off` state of `hint` mode. If `hint` mode is `on`, the menu interface should automatically highlight the gems with valid moves and print out a list of the coordinates together with its valid movement directions.
`Move the Gem!`	Move a gem in a chosen direction. *Note that the related class method will only be implemented in the **next task**. For the current menu, you only need to take in user input for `row`, `col` and `direction`, but no further action needs to be taken.*
`New Game`	Start a new game and reset `hint` mode to be off. For this task, you may just initialize the new game using the `test_board` given.
`Exit`	Exit program.

Evidence 26
Program code of menu implementation. [10]

Task 4.3
Update the class Begemed with the following methods. Note that this task is time consuming and only worth **2 marks**.

<table>
<tr><th>Methods</th><th>Specification</th></tr>
<tr><td>new_game(n: int)</td><td>new_game will now take in a size of n and randomly generate a n x n board of gems. The newly generated gems should not have any connection of 3 or more gems with the same type.</td></tr>
<tr><td>move_gem(row: int, col: int, direction: string)</td><td>move_gem should take in a gem position and direction. If the swap is a valid move, detect any newly formed connection of 3 or more gems with the same type and cancel them.

<table>
<tr><td>

```
     0   1   2   3   4
   +---+---+---+---+---+
0  | r | r | d | e | d |
   +---+---+---+---+---+
1  | s | d | r | d | t |
   +---+---+---+---+---+
2  | d | d | r | s | d |
   +---+---+---+---+---+
3  | e | r | t | r | r |
   +---+---+---+---+---+
4  | t | e | e | s | d |
   +---+---+---+---+---+
```

</td><td>

```
     0   1   2   3   4
   +---+---+---+---+---+
0  | r | r | d | e | d |
   +---+---+---+---+---+
1  | s | d | _ | d | t |
   +---+---+---+---+---+
2  | d | d | _ | s | d |
   +---+---+---+---+---+
3  | e | t | _ | _ | _ |
   +---+---+---+---+---+
4  | t | e | e | s | d |
   +---+---+---+---+---+
```

</td></tr>
<tr><td>Swap "r" at (3, 1) with "t" at (3, 2)</td><td>Gems connected with 3 or more of the same type are cancelled.</td></tr>
</table>

After the gems are being cancelled, those gems on top of the current gems should “fall” down.
New gems will be randomly generated to fill up the board.

<table>
<tr><td>

```
     0   1   2   3   4
   +---+---+---+---+---+
0  | r | r | _ | _ | _ |
   +---+---+---+---+---+
1  | s | d | _ | e | d |
   +---+---+---+---+---+
2  | d | d | _ | d | t |
   +---+---+---+---+---+
3  | e | t | d | s | d |
   +---+---+---+---+---+
4  | t | e | e | s | d |
   +---+---+---+---+---+
```

</td><td>

```
     0   1   2   3   4
   +---+---+---+---+---+
0  | r | r | r | s | s |
   +---+---+---+---+---+
1  | s | d | t | e | d |
   +---+---+---+---+---+
2  | d | d | t | d | t |
   +---+---+---+---+---+
3  | e | t | d | s | d |
   +---+---+---+---+---+
4  | t | e | e | s | d |
   +---+---+---+---+---+
```

</td></tr>
<tr><td>Gems on top of the cancelled gem “falls” down.</td><td>New gems are generated</td></tr>
</table>

Check if there are more connections of 3 or more gems of the same type are formed, if found repeat the above cancel and refill steps.</td></tr>
<tr><td>menu</td><td>Adjust the menu to accommodate these new changes.</td></tr>
</table>

Evidence 27
Program code of above mentioned changes. [2]