

**International School of Engineering**  
**Faculty of Engineering Chulalongkorn University**  
**Course ID: 2190101 Course Name: Computer Programming**  
**First Semester, Final Exam, Date: Friday, December 2<sup>nd</sup>, 2022 Time: 1PM–4PM**

---

Name..... Student ID. ....No. in CR .....

**Instructions**

1. There are 4 parts (40%) in this exam.
2. It is an open-book exam, so you are allowed to open books, notes, labs, files, and videos. However, personal electronic devices (mobile, tablet, laptop) and the internet are not allowed.
3. Once the exam starts, mobile and thumb drives are not allowed to be with you. We cannot have your own belongings; thus, you need to bring a bag to keep your own belongings. If they are found with you during the exam, it is considered cheating resulting in F.
4. You understand and acknowledge the full exam procedure [\[link\]](#).
5. Borrowing is not allowed unless it is supervised by the proctor.
6. You must not bring or send out any part of this exam paper outside. The exam is a government's property. Violators will be prosecuted under a criminal court.
7. Student who wishes to leave the exam room before the end of the exam period, must raise their hand and ask for permission before leaving the room. Student must leave the room in the orderly manner.
8. Once the time is expired, student must stop writing and must remain seated quietly until the proctors collect all the exam papers or given exam booklets. Only then, the students will be allowed to leave the room in the orderly manner.
9. Any student who does not obey the regulations listed above will receive punishment under the Faculty of Engineering Official Announcement on January 6, 2003 regarding the exam regulations.
  - a) **With implicit evidence or showing intention for cheating, student will receive an F in that subject and will receive an academic suspension for 1 semester.**
  - b) **With explicit evidence for cheating, student will receive an F in that subject and will receive an academic suspension for 1 year.**

I acknowledge all instructions above. This exam represents **only my own work**. I did not give or receive help on this exam.

Signature .....

Date .....



## Instruction

- Create a folder “2190101\_Final\_StudentID\_FirstName” on Desktop, e.g.,  
2190101\_Final\_6131755021\_Nuttanan (change to be your student id)
- For each question, save your source code as file.py and rename it as an example below for Final Part1-Part4 (change to be your student id)
  - Final\_Part1\_6131755021.py
  - Final\_Part2\_6131755021.py
  - Final\_Part3-Q1\_6131755021.py
  - Final\_Part3-Q2\_6131755021.py
  - Final\_Part3-Q3\_6131755021.py
  - Final\_Part4\_6131755021.py
- Put all files into your folder

## **Remarks**

- The program that cannot be run cannot be graded resulting in 0 point.
- If you don't strictly follow the above instructions, it can cause your score to be 0 point since our scoring program may not be able to parse and detect your file.

## Part1: Consensus motif

Objective: Basic dictionary and nested structure

Given files: `Final_Part1_toStudent.py`

### Task 1: Write a function `get_consensus(ss)`

Given a set of aligned strings (`ss`) with the same length as shown below and each string is separated by a '\n'.

```
ATCGATGC
GTACaCGC
ACTACAGC
CtCAATTC
CTGgATTC
```

This function returns the most common character(s) in each position. For example, from the above input `ss`, function `get_consensus(ss)` should return the consensus string as shown below

```
A/C T C A/G A T G C
```

where `A/C` represents that A and C are most common in the first position (in this case the characters need to sort alphabetically), T is most common in the second position, C is most common in the third position, and so on.

Hint: The above expected output comes from the counted frequency of each character in each position as shown below.

```
A: 2  0  1  2  4  1  0  0
C: 2  1  2  1  1  1  0  5
G: 1  0  1  2  0  0  3  0
T: 0  4  1  0  0  3  2  0
```

Note that the number of strings in `ss` can vary but all strings will contain the same number of characters. Also, there are only 4 possible characters, i.e., A, T, C or G, in each position, and there is no difference between the small and capital characters. The function will return the output string as capital letters.

## Example Task 1

Function call	Output returned from the function as a string
<pre>ss = ("ATCGATGC\n" +       "GTACaCGC\n" +       "ACTACAGC\n" +       "CtCAATTC\n" +       "CTGgATTc") print(get_consensus(ss))</pre>	A/C T C A/G A T G C
<pre>ss = ("ATCGATGC\n" +       "GTACaCCC\n" +       "ACTACATC\n" +       "CtCAATAC") print(get_consensus(ss))</pre>	A T C A A T A/C/G/T C

### Task 2: Write a function `get_consensus_generic(ss)`

Given a set of aligned strings (`ss`) with the possibly varied length as shown below and each string is separated by a `'\n'`.

```
YCXBATGZ  
YCVBaCGZHQ  
ACXBctGZHQ  
CtCAAtTZHD  
CTXAATTMFD
```

This function returns the most common character(s) in each position as before. For example, from the above input `ss1`, function `get_consensus_generic(ss)` should return the consensus string as shown below

```
C/Y C X B A T G Z H D/Q
```

where C/Y represents that C and Y are most common in the first position, C is most common in the second position and so on. **The difference from the first task is that no-predefined set of characters in each position and the number of characters of each string can vary.** Again, the expected output will always printout the capital letters. Note that you can add your own helper function if you want.

## Example Task 2

Function call	Output returned from the function as a string
<pre>ss = ("YCXBATGZ\n" +       "YCVBaCGZ\n" +       "ACXBCTGZ\n" +       "CtCAATTZ\n" +       "CTXAATTM") print(get_consensus_generic(ss))</pre>	C/Y C X B A T G Z
<pre>ss = ("YCXBATGZ\n" +       "YCVBaCGZHQ\n" +       "ACXBCTGZHQ\n" +       "CtCAATTZHD\n" +       "CTXAATTMFD") print(get_consensus_generic(ss))</pre>	C/Y C X B A T G Z H D/Q
<pre>ss = ("YCXBATGZ\n" +       "YCVCaCGZHQ\n" +       "ACXDCTGZHQ\n" +       "CtCTATTZHD\n" +       "CTXAATTMFD") print(get_consensus_generic(ss))</pre>	C/Y C X A/B/C/D/T A T G Z H D/Q

## Part2: Movie Recommender

Objective: Tuple/Set/Dict

Given files: Final\_Part2\_toStudent.py

Write a program to recommend a movie by watching. The recommend score is based on Jacard similarity that is an intersect over union.

```
def calculate_user_scores(query, users):  
    # query: a dictionary of watching histories for a specific user  
    # users: a dictionary of watching histories for all users  
    # Return: a dictionary of all user's scores (round 2 decimal points)  
  
    user_scores = dict()  
    # fill your code here!  
    return user_scores  
  
def recommend_movies(query, users, user_scores):  
    # query: a dictionary of watching histories for a specific user  
    # users: a dictionary of watching histories for all users  
    # user_scores: a dictionary of all user's scores (round 2 decimal points)  
    # Return: a list of recommend movies in alphabetically order  
  
    recommend = list()  
    # fill your code here!  
    return recommend  
  
def main():  
    # Already provided  
  
main()
```

Users	Watched movies
Pop	"minion", "spiderman"
Tim	"ju-on", "minion"
Pun	"minion"
Puk	"avenger", "batman", "spiderman"
Tan	"spiderman"

Based on the example in the table above,

- If the user (query) watched "minion", we should recommend him/her to watch following users "Pop" and "Tim". So, the recommended movies are ["ju-on", "spiderman"] – **alphabetically order**.
  - $\text{user\_scores}[\text{"Pop"}] = 1/2 = 0.5$
  - $\text{user\_scores}[\text{"Tim"}] = 1/2 = 0.5$
  - $\text{user\_scores}[\text{"Pun"}] = 0$  since already watched all movies ("minion")
  - $\text{user\_scores}[\text{"Puk"}] = 0/4 = 0$
  - $\text{user\_scores}[\text{"Tan"}] = 0/2 = 0$
- If the user (query) watched {"spiderman", "avenger"}, we should recommend him/her to watch following users "Puk". So, the recommended movie is ["batman", "minion"].
  - $\text{user\_scores}[\text{"Pop"}] = 1/3 = 0.33$
  - $\text{user\_scores}[\text{"Tim"}] = 0/4 = 0$
  - $\text{user\_scores}[\text{"Pun"}] = 0/3 = 0$
  - $\text{user\_scores}[\text{"Puk"}] = 2/3 = 0.67$
  - $\text{user\_scores}[\text{"Tan"}] = 0$  since already watched all movies ("spiderman")
- If the user (query) watched {"XXX"}, it should return "No recommend" since it does not intersect with any users.
- If the user (query) watched {} (empty set), it should return "No recommend" since it does not intersect with any users.

## Input

- An integer to get the number of watched movies
- A movie name **(you must lower it)**

## Output

- The result from the calculate\_user\_scores function, which is a dictionary of all user's scores (round 2 decimal points)
- The result from the recommend\_movies function, which is a list of recommended movies with alphabetically order. If no recommended movies, return "No recommendation".

## Example

Input (from keyboard)	Output (on screen)
1 Minion	{'Pop': 0.5, 'Tim': 0.5, 'Pun': 0, 'Puk': 0, 'Tan': 0} ['ju-on', 'spiderman']
2 Spiderman AVENGER	{'Pop': 0.33, 'Tim': 0, 'Pun': 0, 'Puk': 0.67, 'Tan': 0} ['batman']
1 Xxx	{'Pop': 0, 'Tim': 0, 'Pun': 0, 'Puk': 0, 'Tan': 0} No recommendation
0	{'Pop': 0, 'Tim': 0, 'Pun': 0, 'Puk': 0, 'Tan': 0} No recommendation



## Part3:

### Objective: Numpy

Given files: Final\_Part3-Q1\_toStudent.py, Final\_Part3-Q2\_toStudent.py, Final\_Part3-Q3\_toStudent.py

1. (8 points) Write function f1(a) in file final\_Part3-Q1\_toStudent.py

This function receives parameter

- a: a 2-dimensional numpy array that can always be divided into 9 regions, each region with equal number of values (you do not need to check its size).
  - The dimension can be 3x3, 3x6, 3x9, ..., 6x3, 6x6, 6x9, ... etc.

The regions are identified as shown below:

A	B	C
D	E	F
G	H	I

f1 performs element-wise operations of D – F. It returns D-F.

**LOOPS are not allowed in this question! If you use loop, your code will not be marked.**

### Example

Array a						Return value																																					
<table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>77</td><td>5</td><td>13</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></table>						1	2	3	77	5	13	7	8	9	[[64]]																												
1	2	3																																									
77	5	13																																									
7	8	9																																									
<table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td></tr><tr><td>32</td><td>22</td><td>15</td><td>16</td><td>17</td><td>18</td></tr><tr><td>56</td><td>11</td><td>21</td><td>22</td><td>66</td><td>24</td></tr><tr><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td></tr><tr><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td></tr></table>						1	2	3	4	5	6	7	8	9	10	11	12	32	22	15	16	17	18	56	11	21	22	66	24	25	26	27	28	29	30	31	32	33	34	35	36	[[ 15 4] [-10 -13]]	
1	2	3	4	5	6																																						
7	8	9	10	11	12																																						
32	22	15	16	17	18																																						
56	11	21	22	66	24																																						
25	26	27	28	29	30																																						
31	32	33	34	35	36																																						
<table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr><tr><td>10</td><td>21</td><td>43</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>33</td></tr><tr><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td></tr></table>						1	2	3	4	5	6	7	8	9	10	21	43	13	14	15	16	17	33	19	20	21	22	23	24	25	26	27	[[-6 4 10]]										
1	2	3	4	5	6	7	8	9																																			
10	21	43	13	14	15	16	17	33																																			
19	20	21	22	23	24	25	26	27																																			

<table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>1</td><td>8</td><td>10</td></tr><tr><td>66</td><td>11</td><td>22</td></tr><tr><td>13</td><td>14</td><td>15</td></tr><tr><td>16</td><td>17</td><td>18</td></tr></table>	1	2	3	4	5	6	1	8	10	66	11	22	13	14	15	16	17	18	<div>[[ -9] [44]]</div>																																																						
1	2	3																																																																							
4	5	6																																																																							
1	8	10																																																																							
66	11	22																																																																							
13	14	15																																																																							
16	17	18																																																																							
<table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td></tr><tr><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td></tr><tr><td>23</td><td>80</td><td>3</td><td>22</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td></tr><tr><td>91</td><td>37</td><td>66</td><td>88</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td><td>47</td><td>48</td></tr><tr><td>49</td><td>50</td><td>51</td><td>52</td><td>53</td><td>54</td><td>55</td><td>56</td><td>57</td><td>58</td><td>59</td><td>60</td></tr><tr><td>61</td><td>62</td><td>63</td><td>64</td><td>65</td><td>66</td><td>67</td><td>68</td><td>69</td><td>70</td><td>71</td><td>72</td></tr></table>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	23	80	3	22	29	30	31	32	33	34	35	36	91	37	66	88	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	<div>[[ -10 46 -32 -14] [ 46 -9 19 40]]</div>
1	2	3	4	5	6	7	8	9	10	11	12																																																														
13	14	15	16	17	18	19	20	21	22	23	24																																																														
23	80	3	22	29	30	31	32	33	34	35	36																																																														
91	37	66	88	41	42	43	44	45	46	47	48																																																														
49	50	51	52	53	54	55	56	57	58	59	60																																																														
61	62	63	64	65	66	67	68	69	70	71	72																																																														
<table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td></tr><tr><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td></tr><tr><td>11</td><td>22</td><td>21</td><td>22</td><td>23</td><td>24</td></tr><tr><td>33</td><td>38</td><td>27</td><td>28</td><td>29</td><td>30</td></tr><tr><td>21</td><td>45</td><td>33</td><td>34</td><td>35</td><td>36</td></tr><tr><td>37</td><td>38</td><td>39</td><td>40</td><td>41</td><td>42</td></tr><tr><td>43</td><td>44</td><td>45</td><td>46</td><td>47</td><td>48</td></tr><tr><td>49</td><td>50</td><td>51</td><td>52</td><td>53</td><td>54</td></tr></table>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	11	22	21	22	23	24	33	38	27	28	29	30	21	45	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	<div>[[ -12 -2] [ 4 8] [-14 9]]</div>																		
1	2	3	4	5	6																																																																				
7	8	9	10	11	12																																																																				
13	14	15	16	17	18																																																																				
11	22	21	22	23	24																																																																				
33	38	27	28	29	30																																																																				
21	45	33	34	35	36																																																																				
37	38	39	40	41	42																																																																				
43	44	45	46	47	48																																																																				
49	50	51	52	53	54																																																																				

2. (5 points) Write function f2(a) in file final\_Part3-Q2\_toStudent.py

This function receives parameter

- a: a two-dimensional numpy array containing strings.

Function f2 returns a new array that is the result from

- slicing array a, such that it takes only even rows and odd columns.
- And then make the result 1-dimension numpy array.

**LOOPS are not allowed in this question! If you use loop, your code will not be marked.**

## Example

Array a					Return value
e	f	g			[['f','i','x']]
a	b	c			
h	i	j			
q	r	s			
w	x	y			
m	l	n	o	t	[['l' 'o' 'i' 'd' 'a' 's' 'p' 'y']]
e	r	h	u	n	
t	i	r	d	o	
m	e	a	l	o	
n	a	w	s	y	
f	r	o	m	b	
e	p	n	y	s	

1	l	d	u	6	m	s	i
8	h	n	u	n	u	n	b
t	s	2	1	o	s	4	t
m	e	a	k	o	f	o	v
n	w	w	i	3	f	y	u
f	r	5	m	b	m	b	m

```
[[ 'l' 'u' 'm' 'i' 's' 'l' 's'
  't' 'w' 'i' 'f' 'u' ]]
```

3. (7 points) Write 2 functions in file final\_Part3-Q3\_toStudent.py

A (simplified) football league table looks like:

Number of win, draw, and lost matches						Win/not win in the last 5 matches (1 = win, 0 = not win)				
	Win	Draw	Lose	GoalFor	GoalAgainst	match1	match2	match3	match4	match5
Arsenal	11	1	1	31	11	1	1	0	1	1
ManCity	10	2	1	39	12	1	1	1	0	1
ManU	7	2	4	18	19	0	1	0	1	0
Chelsea	6	3	4	17	16	0	0	0	0	1

Team names

This is array a

Write the following functions

- (4 points) points(names,a): This function receives parameters:
  - names: a numpy array containing names of football teams.
  - a: 2-dimensional numpy array of football league table (not including column names and row names).

Assume that the number of teams are equal to the number of rows in array a, this function creates and returns an array that represents all points each team received (3 points for each win, 1 point for each draw, 0 point for each loss). For the array in the picture, the returned array will be

```
[[34]
 [32]
 [23]
 [21]]
```

Where 34 comes from  $3 \times 11 + 1 \times 1 + 0 \times 1$

**LOOPS are not allowed in part a)! If you use loop, your code will not be marked.**

- (3 points) reportPoints(names,ptsTotal): This function receives parameters:
  - names: an array containing names of football teams.

ii. ptsTotal: the point array calculated from a).

Assume that the number of teams are equal to the number of rows in array a, this function creates and returns a string that reports the points of all teams. For the array, a, in the picture, the returned string will be

Arsenal 34

ManCity 32

ManU 23

Chelsea 21

(The actual string is given in the file).

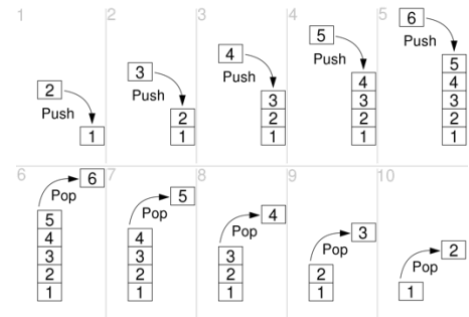
## Part4: Stack

Objective: Class & Object

Given files: Final\_Part4\_toStudent.py

**Stack** is an abstract data type that serves as a collection of elements. There are two main operations:

- **push**, which adds an element to collection and,
- **pop**, which removes the most recently added element that was not yet removed.



The order in which an element added to or removed from a stack is described as last in, first out, referred to by the acronym LIFO.

Considered as a **linear data structure**, or more abstractly a sequential collection, the **push** and **pop** operations occur only at one end of the structure, referred to as the top of the stack. A stack may be implemented to have a bounded capacity. If the stack is full and does not contain enough space to accept another element, the stack is in a state of stack overflow.

You are required to implement this stack class and function **main()** (**total of 5 tasks**) as the following specifications:

Stack's properties:

- **capacity** (*an integer number*), the bounded capacity of the stack
- **content** (*a list with the number of elements equal to size*), the collection of elements stored in stack
- **tos** (*an integer number*), the top of the stack location in content

Methods:

#### Task 1

- **\_\_init\_\_(self, size)**, a constructor, initializes all stack's properties ([see input 1 in example](#))
  - **capacity** should be initialized to the value of **size**
  - **content** should be initialized to a list of  $n$  elements where  $n$  is the capacity of the stack. Initially, the content stores **None** for all elements.
  - **tos** should be initialized to the location that the first element can be add to stack, index 0 of content.

#### Task 2

- **\_\_str\_\_(self)**, returns a string representation of the stack which shows the properties of the stack. The table below shows example of string representation of a stack of capacity 3, '\*' denotes the top of the stack location. ([see input 2 in example](#))

Stack is empty	Stack has some data	Stack is full
----- None None None* -----	----- None None* 1 -----	-----* 4 2 1 -----

#### Task 3

- **push(self)**, adds an element at the top of the stack, the top of the stack must be changed to new location. If the stack is full, nothing should be add to the stack and an error message '**ERROR: stack is full**' must be printed on screen. This method returns nothing. ([see input 3 in example](#))

#### Task 4

- **pop(self)**, removes the most recently added element that was not yet removed and return the removed element. If the stack is empty, an error message '**ERROR: stack is empty**' must be printed on screen and nothing will be returned. ([see input 4 in example](#))

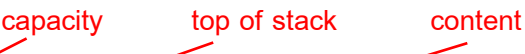

#### Task 5

In addition, complete the function **main()** to show that you can use stack properly. The steps in **main()** are provided as following: ([see input 5 in example](#))

1. Get an integer number, **n**
2. Create an empty stack with capacity equal **n + 1**, and print the stack created
3. Get **n** integer numbers from keyboard one line at time. Each time the input is entered the data should be added to stack created in (2). Print the stack after all data are added

4. Remove all elements except the bottom most of the stack, calculate the summation of removed elements and print the summation on screen
5. Print the stack (DO NOT use show\_stack())

## Example

Input (from keyboard)	Output (on screen)
1  <code>test_init()</code>	 <pre> Empty stack capacity: 2 2 0 [None, None] Empty stack capacity: 4 4 0 [None, None, None, None] Empty stack capacity: 5 5 0 [None, None, None, None, None] </pre>
2  <code>test_str()</code>	<pre> stack is empty ----- None None None* ----- stack has element ----- None None* 1 ----- stack is full -----* 4 2 1 ----- </pre>
3  <code>test_push()</code>	<pre> 3 initial 3 0 [None, None, None] ----- s.push(2) return from push: None 3 1 [2, None, None] ----- s.push(4) return from push: None 3 2 [2, 4, None] ----- s.push(5) return from push: None 3 3 [2, 4, 5] ----- s.push(6) ERROR: stack is full return from push: None 3 3 [2, 4, 5] ----- </pre>
4  <code>test_pop()</code>	 <pre> 3 3 [2, 4, 5] -&gt; 5 3 2 [2, 4, None] -&gt; 4 3 1 [2, None, None] -&gt; 2 -&gt; None 3 0 [None, None, None] ERROR: stack is empty -&gt; None 3 0 [None, None, None] </pre>

