ให้น้องๆเขียนโปรแกรมโดยรับ input 2 แบบ โดยใช้ QUEUE ในการแก้ปัญหา

E  <value>  ให้นำ value ไปใส่ใน QUEUE และทำการแสดงผลค่าที่ทำการ enqueue และ index ของตัวที่ทำการเพิ่มเข้าไป

D                 ให้ทำการ dequeue ตัวที่อยู่หน้าสุดของ Queue ออกและแสดงตัวเลขที่เอาออกและแสดงขนาดของ Queue

                    หลังจากทำการ dequeue แล้ว

\*\*\* ในตอนสุดท้ยถ้าหากใน Queue ยังมี Value อยู่ให้แสดงผลออกมา  ถ้าหากไม่มีแล้วให้แสดงคำว่า  Empty \*\*\*

Enter Input : E 10,E 20,E 30,E 40,D,D

Add 10 index is 0

Add 20 index is 1

Add 30 index is 2

Add 40 index is 3

Pop 10 size in queue is 3

Pop 20 size in queue is 2

Number in Queue is : ['30', '40']

Enter Input : E 10,E 20,E 30,E 40,D,E 50,E 60,D,D,D,D,D,D

Add 10 index is 0

Add 20 index is 1

Add 30 index is 2

Add 40 index is 3

Pop 10 size in queue is 3

Add 50 index is 3

Add 60 index is 4

Pop 20 size in queue is 4

Pop 30 size in queue is 3

Pop 40 size in queue is 2

Pop 50 size in queue is 1

Pop 60 size in queue is 0

-1

Empty

Enter Input : D,E 99,D,D,E 88,D,D,E 12,E 13,E 86

-1

Add 99 index is 0

Pop 99 size in queue is 0

-1

Add 88 index is 0

Pop 88 size in queue is 0

-1

Add 12 index is 0

Add 13 index is 1

Add 86 index is 2

Number in Queue is : ['12', '13', '86']

Enter Input : D,D,D,D,D

-1

-1

-1

-1

-1

Empty

-ข้อ 2 Q

PSD48 (P-Saderd 48 Group) เป็นวงไอดอลวงหนึ่งที่กระแสกำลังมาแรง ณ ตอนนี้โดยเพลงที่ได้รับความนิยมอย่างมากคือเพลงจี่หอย โดยวง PSD48 กำลังจะจัดงานจับมือขึ้น โดยมีกฎอยู่ว่าถ้าหากคนที่กำลังต่อแถวอยู่เป็นคนจาก กองกำลังสำรวจ จะได้สิทธิพิเศษในการแทรกแถวไปข้างหน้าสุดทันที (แต่ถ้าหากคนหน้าสุดก็เป็นคนของกองกำลังสำรวจก็ต้องต่อหลังเขาอยู่ดี)  PSD48 อยากให้คุณช่วยเขียนโปรแกรมสำหรับหาว่าจะมีโอตะ id ใดบ้างที่ได้จับมือ

เพลงประกอบ : https://youtu.be/Jd4Hd-HFgls

Input :  
EN <value>  เป็นโอตะธรรมดา  id = value  
ES <value>  เป็นโอตะของกองกำลังสำรวจ  id = value  
D                  เป็นคำสั่งแสดงผล value ของหัวแถว ถ้าหากในแถวไม่มีคนจะแสดงคำว่า Empty

Enter Input : EN 1,EN 2,D,D,D,EN 3,D

1

2

Empty

3

Enter Input : EN 1,ES 2,D,D,D,EN 3,D

2

1

Empty

3

Enter Input : EN 1,ES 2,ES 99,D,D,D,EN 3,D

2

99

1

3

-ข้อ 3 Q

จงเขียน ฟังก์ชั่นสำหรับการ encode และ decode ของ String ที่รับมาโดยให้ทำเป็นรูปแบบ Queue

รูปแบบการรับ Input โดยจะคั่นแต่ละตำแหน่งด้วย คอมม่า(',') :

    - ตำแหน่งที่หนึ่ง string ไม่จำกัดความยาวโดยที่จะไม่นับช่องว่าง(spacebar)

    - ตำแหน่งที่สอง ชุดตัวเลข(1-9)

โดยที่รูปแบบการ encode คือ การนำ String ที่รับมาเพิ่มค่า ascii เท่ากับค่าของชุดตัวเลขในตำแหน่งแรก หลังจากนั้นให้ dequeue ชุดตัวเลขออกไปไว้ข้างหลังสุด เช่น ตัวอักษรตำแหน่งแรกคือ i และชุดตัวเลขในตำแหน่งแรกคือ 2 ดังนั้นตัวอักษรที่ได้จากการ encode คือ k โดยจะทำการวนชุดตัวเลขไปเรื่อยๆจนกระทั่งทำการ encode ทุกตัวอักษรใน String ครบ ถ้าหากผลลัพธ์จากการเพิ่มหรือลดค่า ascii ไม่ใช่ตัวอักษรให้กลับมาวนในชุดตัวอักษร เช่น อักษร z ทำการ encode ด้วยเลข 2 จะได้ b และหากทำการ decode ตัวอักษร A ด้วย 2 จะได้ Y

โดยการ decode หลังจาก encode ต้องให้คำตอบที่มีค่าเท่ากับ String เดิมก่อนทำการ encode

\*\*\*ให้ใช้วิธี enqueue และ dequeue ในการเลื่อนตำแหน่ง เท่านั้น\*\*\*

โดยรูปแบบการ run ดังนี้ :

q1 = Queue(string)

q2 = Queue(number)

encodemsg(q1, q2)

decodemsg(q1, q2)

Enter String and Code : i love Python,256183

Encode message is : ['k', 'q', 'u', 'w', 'm', 'S', 'a', 'y', 'n', 'p', 'v']

Decode message is : ['i', 'l', 'o', 'v', 'e', 'P', 'y', 't', 'h', 'o', 'n']

Enter String and Code : KMITL,2

Encode message is : ['M', 'O', 'K', 'V', 'N']

Decode message is : ['K', 'M', 'I', 'T', 'L']

Enter String and Code : zzzzzzzzz,123456789

Encode message is : ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']

Decode message is : ['z', 'z', 'z', 'z', 'z', 'z', 'z', 'z', 'z']

Enter String and Code : I Love Data Structures But I Hate Bug,999997

Encode message is : ['R', 'U', 'x', 'e', 'n', 'K', 'j', 'c', 'j', 'B', 'c', 'y', 'd', 'l', 'c', 'd', 'a', 'l', 'b', 'K', 'd', 'c', 'R', 'O', 'j', 'c', 'n', 'K', 'd', 'n']

Decode message is : ['I', 'L', 'o', 'v', 'e', 'D', 'a', 't', 'a', 'S', 't', 'r', 'u', 'c', 't', 'u', 'r', 'e', 's', 'B', 'u', 't', 'I', 'H', 'a', 't', 'e', 'B', 'u', 'g']

Enter String and Code : SawaddeeKub,55555

Encode message is : ['X', 'f', 'b', 'f', 'i', 'i', 'j', 'j', 'P', 'z', 'g']

Decode message is : ['S', 'a', 'w', 'a', 'd', 'd', 'e', 'e', 'K', 'u', 'b']

ข้อ 4 Q

สมมติว่านักศึกษาแอบชอบคนๆหนึ่งอยู่ โดยที่นักศึกษาและคนๆนั้นจะมีกิจกรรมและสถานที่ที่ไปแตกต่างกันในแต่ละวัน  
ให้นักศึกษาเขียนโปรแกรมที่จะหาว่าสิ่งที่นักศึกษาและคนๆนั้นทำในแต่ละวันจะทำให้ได้คบกันหรือไม่ ***โดยใช้ Queue***

**กิจกรรม**                                       **สถานที่**  
0 = กินข้าว(Eat)                           0 = ร้านอาหาร(Res.)  
1 = เล่นเกม(Game)                      1 = ห้องเรียน(ClassR.)  
2 = ทำโจทย์ datastruc(Learn)      2 = ห้างสรรพสินค้า(SuperM.)  
3 = ดูหนัง(Movie)                        3 = บ้าน(Home)

**โดยการรับ Input จะประกอบด้วย**

กิจกรรม:สถานที่(ของนักศึกษาและของคนๆนั้น) โดยในแต่ละวันจะคั่นด้วยเครื่องหมาย ,

เช่น วันที่ 1 นักศึกษาไปกินข้าวที่ร้านอาหาร และ คนๆนั้นไปนั่งทำโจทย์ datastruc ที่ร้านอาหาร   
       วันที่ 2 นักศึกษาไปเล่นเกมที่บ้าน และ คนๆนั้นไปดูหนังที่ห้างสรรพสินค้า  
จะได้ว่า 0:0 2:0,1:3 3:2

**\*\*\*มีการคิดคะแนนดังนี้\*\*\***

·       กิจกรรมเดียวกันแต่คนละสถานที่         +1

·       สถานที่เดียวกันแต่ทำกิจกรรมต่างกัน    +2

·       กิจกรรมเดียวกันและสถานที่เดียวกัน    +4

·       ไม่เหมือนกันเลย                                   - 5

หากมีคะแนนมากกว่าหรือเท่ากับ 7 จะถือว่าได้คบกัน แต่ถ้าคะแนนน้อยกว่า 7 แต่มากกว่า 0 เป็นคนคุย น้อยกว่านั้นถือว่าเป็นได้แค่เพื่อน

โดยในแต่ละขั้นตอนให้แสดงผลดังตัวอย่าง

Enter Input : 0:0 2:0,1:3 3:3,2:1 2:1

My Queue = 0:0, 1:3, 2:1

Your Queue = 2:0, 3:3, 2:1

My Activity:Location = Eat:Res., Game:Home, Learn:ClassR.

Your Activity:Location = Learn:Res., Movie:Home, Learn:ClassR.

Yes! You're my love! : Score is 8.

Enter Input : 2:1 2:1

My Queue = 2:1

Your Queue = 2:1

My Activity:Location = Learn:ClassR.

Your Activity:Location = Learn:ClassR.

Umm.. It's complicated relationship! : Score is 4.

Enter Input : 0:1 2:3,3:2 3:2

My Queue = 0:1, 3:2

Your Queue = 2:3, 3:2

My Activity:Location = Eat:ClassR., Movie:SuperM.

Your Activity:Location = Learn:Home, Movie:SuperM.

No! We're just friends. : Score is -1.

Enter Input : 3:3 1:3,0:0 1:1,2:2 3:3,0:3 0:1

My Queue = 3:3, 0:0, 2:2, 0:3

Your Queue = 1:3, 1:1, 3:3, 0:1

My Activity:Location = Movie:Home, Eat:Res., Learn:SuperM., Eat:Home

Your Activity:Location = Game:Home, Game:ClassR., Movie:Home, Eat:ClassR.

No! We're just friends. : Score is -7.

ข้อ 5 Q

**เกม Color Crush คืออะไร** : Color Crush จะเป็นเกมที่นำสีมาเรียงต่อกัน โดยสีจะหายไปก็ต่อเมื่อมีการเรียงสีเหมือนกันครบ 3 อัน เช่น  ABBBA  -> AA  เนื่องจาก B เรียงติดกัน 3 ตัวทำให้ระเบิดหายไปโดยที่สีจะมีทั้งหมด 26 สี และจะถูกแทนด้วย A - Z  ถ้าหากมีการเรียงกันแบบ  ABBBAA -> Empty  เนื่องจาก  ถ้าหาก B ระเบิด  A(BBB)AA -> AAA จะเห็นว่า A ก็เรียงกันอีก 3 ตัวทำให้เกิดการระเบิดขึ้นอีกครั้งหนึ่ง  และถ้าหากมีการเรียงกันแบบ AAAA -> A เนื่องจากมีการเรียงกัน 3 ตัว  (AAA)A ทำให้เหลือ A 1 ตัว

**เนื้อเรื่อง** :  หลังจากที่กฤษฎาได้เล่นเกม Color Crush ก็ได้ไปเห็นโฆษณาว่า บริษัทที่ได้สร้าง Color Crush มีแผนการที่จะสร้างเกม Color Crush 2 ขึ้นมา กฤษฎาจึงได้สมัครเข้าไปร่วมทีมในการสร้างเกม Color Crush 2 ซึ่งเกมนี้จะมีกิมมิคที่แตกต่างออกไป คือการที่จะมี 2 ฝั่ง คือ ฝั่งปกติกับฝั่งโลกกระจก โดยฝั่งโลกกระจกจะเกิดการระเบิดก่อน ซึ่งการระเบิดของฝั่งโลกกระจกจะไม่ใช่ระเบิดแล้วหายไปเลย แต่จะเป็นระเบิดแล้วกลายเป็น ITEM ไว้สำหรับขัดขวางการระเบิดของฝั่งปกติ  หลังจากที่ฝั่งโลกกระจกเกิดการระเบิดครบแล้ว ก็จะเป็นคิวของฝั่งปกติ  ซึ่งถ้าหากฝั่งปกติมีการเรียงกันของสีที่จะทำให้เกิดการระเบิด ในเสี้ยววินาทีนั้นก่อนที่จะเกิดการระเบิดของฝั่งปกติ  ITEM สำหรับขัดขวางการระเบิดของฝั่งโลกกระจก จะมาคั่นระหว่างระเบิดลูกที่ 2 กับ ลูกที่ 3 (อาจจะทำให้เกิดการระเบิดเหมือนเดิมได้ถ้าหาก ระเบิดนั้นเป็นสีเดียวกัน  แต่ถ้าเป็นคนละสีก็จะทำให้ไม่เกิดการระเบิดขึ้น)  โดยระเบิดอาจจะเกิดการระเบิดซ้อนๆกันเรื่อยๆได้จะเป็น Empty  เช่น ถ้าหากฝั่งปกติมีระเบิดเรียงแบบนี้ AAAAA และฝั่งโลกกระจกมีระเบิดแบบนี้ AAA ถ้าหากฝั่งปกติระเบิดธรรมดา 1 ทีจะเหลือแค่ AA แต่ถ้าหากฝั่งโลกกระจกมาขัดขวาง จะกลายเป็น AA(A)AAAA ก็จะเกิดระเบิด 2 ทีทำให้ระเบิดฝั่งปกติเป็น Empty

**อธิบายรูปแบบ Input ของ Test\_Case\_1** : ฝั่งปกติจะมีระเบิดเรียงดังนี้ -> AAABBBCDEE  ฝั่งโลกกระจกจะมีระเบิดเรียงดังนี้ -> HHH โดยฝั่งโลกกระจกจะมีระเบิด H ที่เป็น ITEM สำหรับขัดขวาง 1 ลูกไว้สำหรับขัดขวางการระเบิดที่จะเกิดขึ้นกับฝั่งปกติได้  ต่อมาฝั่งปกติจะเกิดการระเบิดของ A และ B ตามลำดับ  โดยฝั่งโลกกระจกจะนำระเบิด H ไปขัดขวางการระเบิดของระเบิด A เพราะระเบิด A เกิดการระเบิดก่อนระเบิด B  โดยการขัดระเบิดนั้นจะเป็นการขัดระหว่างลูกที่ 2 กับลูกที่ 3 เพื่อให้เห็นภาพ -> AAABBBCDEE -> AA(H)ABBBCDEE  -> AA(H)ACDEE ลำดับจะเป็นดังนี้  และฝั่งปกติเกิดการระเบิด 1 ครั้ง ส่วนฝั่งโลกกระจกก็เกิดการระเบิดอีก 1 ครั้ง

**อธิบายรูปแบบ Input ของ Test\_Case\_3** : ฝั่งปกติจะมีระเบิดเรียงดังนี้ -> AAABBBCDDDEE  ฝั่งโลกกระจกจะมีระเบิดเรียงดังนี้ -> BBBTENETAAA โดยฝั่งโลกกระจกจะมีระเบิด A และ B ที่เป็น ITEM สำหรับขัดขวาง 2 ลูกตามลำดับไว้สำหรับป้องกันการระเบิดที่จะเกิดขึ้นกับฝั่งปกติได้  ต่อมาฝั่งปกติจะเกิดการระเบิดของ A B และ D ตามลำดับ  โดยฝั่งโลกกระจกจะนำระเบิด A  ไปขัดขวางการระเบิดของระเบิด A เพราะระเบิด A เกิดการระเบิดก่อนระเบิด B  โดยการขัดระเบิดนั้นจะเป็นการขัดระหว่างลูกที่ 2 กับลูกที่ 3  เพื่อให้เห็นภาพ -> AAABBBCDDDEE -> AA(A)ABBBCDDDEE -> ABBBCDDDEE ลำดับจะเป็นดังนี้  ต่อมาจะนำระเบิด B ไปขัดขวางการระเบิดของระเบิด B เพื่อให้เห็นภาพ  ABBBCDDDEE -> ABB(B)BCDDDEE -> ABCDDDEE  ต่อมาเกิดการระเบิดอีก 1 ครั้ง ABCDDDEE -> ABCEE ซึ่งฝั่งโลกกระจกไม่สามารถขัดขวางได้เพราะ ITEM สำหรับขัดขวางหมดแล้ว   และฝั่งปกติเกิดการระเบิดทั้งหมด 3 ครั้ง  ซึ่ง 2 ครั้งเกิดจากการที่ฝั่งโลกกระจกใส่ระเบิดสีเดียวกันมาซึ่งถือว่าเป็นการขัดขวางที่ผิดหและเกิดการระเบิดเองอีก 1 ครั้ง ส่วนฝั่งโลกกระจกก็เกิดการระเบิดอีก 2 ครั้ง

**อธิบายรูปแบบ Output** : แบ่งออกเป็น 2 ฝั่งคือฝั่งปกติกับฝั่งโลกกระจก  โดยบรรทัดแรกจะเป็นจำนวนระเบิดที่เหลืออยู่ บรรทัดที่สองจะเป็นระเบิดที่เหลืออยู่แต่ถ้าหากไม่มีระเบิดเหลืออยุ่เลยให้แสดง "Empty" บรรทัดที่สามจะเป็นจำนวนที่เกิดระเบิดขึ้น บรรทัดที่สี่จะมีเฉพาะฝั่งปกติถ้าหากเกิดเหตุการณ์ที่ ITEM ของฝั่งโลกกระจกมาขัดขวาง แต่ระเบิดนั้นดันเป็นลูกเดียวกับที่จะเกิดการระเบิด  ส่วนทีมสีน้ำเงินจะเหมือนกับทีมสีแดงแต่บรรทัดที่ 2 กับ 3 และชื่อทีม จะเป็นแบบ inverse

**คำใบ้** - ใช้ Stack ในการหาลูกระเบิดเรียงกัน 3 ลูก   โดยให้ทำฝั่งโลกกระจกก่อนว่ามีระเบิดลูกอะไรบ้าง (ก่อนเข้า stack ให้ Reverse ก่อน)  จากนั้นเก็บลง Queue แล้วไปทำฝั่งปกติถ้าหากฝั่งปกติเกิดการระเบิดก็ DeQueue ระเบิดที่ได้รับมาจากฝั่งกระจกมาขัดระเบิดระหว่างลูกที่ 2 กับ 3

**อธิบาย Case 10:**

ฝั่งซ้าย = DDDFFFGGG  
ฝั่งขวา = ABBBAACCC  
ทำฝั่งขวาก่อนโดยการ inverse ABBBAACCC -> CCCAABBBA จะได้ระเบิดมา 3 ลูกคือ C B A ตามลำดับจากนั้นเก็บลง Queue ต่อมาดูที่ฝั่งซ้าย DDD จะเกิดการระเบิดเราจะนำ C ไปขัด | ต่อมา F จะระเบิดเราจะนำ B มาขัด | ต่อมา G จะระเบิดเราจะนำ A มาขัด   สุดท้ายจะกลายเป็น DDCDFFBFGGAG

Enter Input (Normal, Mirror) : AAABBBCDEE HHH

NORMAL :

8

EEDCAHAA

1 Explosive(s) ! ! ! (NORMAL)

------------MIRROR------------

: RORRIM

0

ytpmE

(RORRIM) ! ! ! (s)evisolpxE 1

Enter Input (Normal, Mirror) : AAABBBCDEE FGHHHIOPPP

NORMAL :

12

EEDCBHBBAPAA

0 Explosive(s) ! ! ! (NORMAL)

------------MIRROR------------

: RORRIM

4

FGIO

(RORRIM) ! ! ! (s)evisolpxE 2

Enter Input (Normal, Mirror) : AAABBBDDD TENET

NORMAL :

0

Empty

3 Explosive(s) ! ! ! (NORMAL)

------------MIRROR------------

: RORRIM

5

TENET

(RORRIM) ! ! ! (s)evisolpxE 0

Linklist

ข้อ 1

สร้าง method insert ในคลาส LinkedList เพื่อแทรกข้อมูลลงใน index ที่กำหนดของ linked list และ return ผลลัพธ์ตามตัวอย่าง

โดยคลาส LinkedList จะประกอบไปด้วย

1. def \_\_init\_\_(self): สำหรับสร้าง linked list

2. def \_\_str\_\_(self): return string แสดง ค่าใน linked list

3. def isEmpty(self): return list นั้นว่างหรือไม่

4. def append(self, data): เพิ่ม data ต่อท้าย linked list

5. def insert(self, index, data): insert data ใน index ที่กำหนด

โดยการแทรกในที่นี้ จะเป็นการนำข้อมูลใหม่ที่ต้องการมาใส่แทนที่ตำแหน่งของข้อมูลเดิมและย้ายข้อมูลเดิมไปต่อหลังข้อมูลใหม่

**คำแนะนำเพิ่มเติม** เพื่อความง่ายในการเขียนโค้ดและไม่ต้องเขียนspecial caseเยอะๆ ให้ลองใช้ Dummy Node ดูนะครับ(หากสงสัยการใช้งาน Dummy Node สอบถามพี่ๆTA หรือ <https://youtu.be/XgUIjTQ1HjA> )

**\*\*\*\*\*\*\*ให้ใช้ class Node ในการทำ Linked List ห้ามใช้ list\*\*\*\*\*\*\*\*\***

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.next = None

Enter Input : 1 2, 0:0, 3:3

link list : 1->2

index = 0 and data = 0

link list : 0->1->2

index = 3 and data = 3

link list : 0->1->2->3

Enter Input : 0 1 2, -1:3, 10:10

link list : 0->1->2

Data cannot be added

link list : 0->1->2

Data cannot be added

link list : 0->1->2

Enter Input : 0 1 2 4, 3:3

link list : 0->1->2->4

index = 3 and data = 3

link list : 0->1->2->3->4

Enter Input : ,0:0,1:1

List is empty

index = 0 and data = 0

link list : 0

index = 1 and data = 1

link list : 0->1

Enter Input : ,1:1

List is empty

Data cannot be added

List is empty

2.

ให้เขียนคลาสของ Doubly Linked List ซึ่งมีเมท็อดดังนี้

1. \_\_init\_\_     สร้าง Head ขึ้นมาเพื่อบอกว่าจุดเริ่มต้นของ Linked List คือตรงไหน

2. \_\_str\_\_     คืนค่าเป็นสตริงซึ่งบอกว่า Linked List เราตั้งแต่หัวไปจนท้ายมีตัวอะไรบ้าง

3. reverse     คืนค่าเป็นสตริงซึ่งบอกว่า Linked List เราตั้งแต่ท้ายไปจนหัวมีตัวอะไรบ้าง

4. isEmpty    เช็คว่า Linked List ของเราว่างหรือป่าว คืนค่าเป็น True / False

5. append     add Item เข้า Linked List จากด้านหลัง ไม่คืนค่า

6. addHead  add Item เข้า Linked List จากด้านหน้า ไม่คืนค่า

7. search      ค้นหา Item ที่ต้องการใน Linked List คืนค่าเป็น Found / Not Found

8. index        ค้นหา Item ที่ต้องการใน Linked List ว่าอยู่ที่ Index ไหน คืนค่าเป็น Index (0,1,2,3,4,.....) ถ้าหากไม่มีคืนค่าเป็น -1

9. size           คืนค่าเป็นขนาดของ Linked List

10. pop         นำ Item Index ที่ pos ออกจาก Linked List คืนค่าเป็น Success / Out of Range

11. insert       เป็นการนำ Item ไปแทรกใน Linked List ตามตำแหน่ง pos ไม่มีการคืนค่า

ถ้าน้องยังไม่ค่อยเข้าใจการทำงานของ insert ให้น้องลองกับ List บน Python ได้  เช่น

1.  มี arr = [ 0 , 1 , 2 , 3 ] แล้วเรา arr.insert(0,"T") จะได้ผลลัพธ์คือ [ "T", 0 , 1 , 2 , 3 ]

2.  มี arr = [ 0 , 1 , 2 , 3 ] แล้วเรา arr.insert(999,"T") จะได้ผลลัพธ์คือ [ 0 , 1 , 2 , 3 , "T" ]

3.  มี arr = [ 0 , 1 , 2 , 3 ] แล้วเรา arr.insert(-2,"T") จะได้ผลลัพธ์คือ [ 0 , 1 , "T" , 2 , 3 ]

4.  มี arr = [ 0 , 1 , 2 , 3 ] แล้วเรา arr.insert(-10,"T") จะได้ผลลัพธ์คือ [ "T", 0 , 1 , 2 , 3 ]

โดยรูปแบบ Input มีดังนี้

1. append    ->  AP

2. addHead  ->  AH

3. search      ->  SE

4. index        ->   ID

5. size          ->   SI

6. pop          ->   PO

7. insert       ->   IS

โดยให้เพิ่มเติมจากส่วน #Code Here ของโปรแกรมต่อไปนี้ เพื่อให้สามารถแสดงผลได้ตามที่โจทย์กำหนด

\*\*\*\*\*\*\*\*  ห้ามใช้ List ในการทำ Linked List เด็ดขาดถ้าหากพบจะถูกลดเป็น 0 คะแนน \*\*\*\*\*\*\*\*

class Node:

    def \_\_init\_\_(self, value):

        self.value = value

        self.next = None

        self.previous = None

class LinkedList:

    def \_\_init\_\_(self):

        self.head = None

        self.tail = None

    def \_\_str\_\_(self):

        if self.isEmpty():

            return "Empty"

        cur, s = self.head, str(self.head.value) + " "

        while cur.next != None:

            s += str(cur.next.value) + " "

            cur = cur.next

        return s

    def reverse(self):

        if self.isEmpty():

            return "Empty"

        cur, s = self.tail, str(self.tail.value) + " "

        while cur.previous != None:

            s += str(cur.previous.value) + " "

            cur = cur.previous

        return s

    def isEmpty(self):

        return self.head == None

    def append(self, item):

        #Code Here

    def addHead(self, item):

        #Code Here

    def insert(self, pos, item):

        #Code Here

    def search(self, item):

        #Code Here

    def index(self, item):

        #Code Here

    def size(self):

        #Code Here

    def pop(self, pos):

        #Code Here

L = LinkedList()

inp = input('Enter Input : ').split(',')

for i in inp:

    if i[:2] == "AP":

        L.append(i[3:])

    elif i[:2] == "AH":

        L.addHead(i[3:])

    elif i[:2] == "SE":

        print("{0} {1} in {2}".format(L.search(i[3:]), i[3:], L))

    elif i[:2] == "SI":

        print("Linked List size = {0} : {1}".format(L.size(), L))

    elif i[:2] == "ID":

        print("Index ({0}) = {1} : {2}".format(i[3:], L.index(i[3:]), L))

    elif i[:2] == "PO":

        before = "{}".format(L)

        k = L.pop(int(i[3:]))

        print(("{0} | {1}-> {2}".format(k, before, L)) if k == "Success" else ("{0} | {1}".format(k, L)))

    elif i[:2] == "IS":

        data = i[3:].split()

        L.insert(int(data[0]), data[1])

print("Linked List :", L)

print("Linked List Reverse :", L.reverse())

Enter Input : AP I,AP Love,AP KMITL,AP 2020

Linked List : I Love KMITL 2020

Linked List Reverse : 2020 KMITL Love I

Enter Input : PO -999,PO 999,PO 0,AP KMITL,PO 999,PO 0

Out of Range | Empty

Out of Range | Empty

Out of Range | Empty

Out of Range | KMITL

Success | KMITL -> Empty

Linked List : Empty

Linked List Reverse : Empty

Enter Input : SI,AH 2020,SI,AH KMITL,SI,AH LOVE,SI,AH I,SI

Linked List size = 0 : Empty

Linked List size = 1 : 2020

Linked List size = 2 : KMITL 2020

Linked List size = 3 : LOVE KMITL 2020

Linked List size = 4 : I LOVE KMITL 2020

Linked List : I LOVE KMITL 2020

Linked List Reverse : 2020 KMITL LOVE I

Enter Input : SI,IS 0 KMITL,SI,SE 0,SE KMITL,ID 0,ID KMITL,PO 0,SI,IS -999 CE,SI

Linked List size = 0 : Empty

Linked List size = 1 : KMITL

Not Found 0 in KMITL

Found KMITL in KMITL

Index (0) = -1 : KMITL

Index (KMITL) = 0 : KMITL

Success | KMITL -> Empty

Linked List size = 0 : Empty

Linked List size = 1 : CE

Linked List : CE

Linked List Reverse : CE

Enter Input : AP 0,AP 1,AP 2,AP 3,SI,IS -1 KMITL,SI,ID KMITL

Linked List size = 4 : 0 1 2 3

Linked List size = 5 : 0 1 2 KMITL 3

Index (KMITL) = 3 : 0 1 2 KMITL 3

Linked List : 0 1 2 KMITL 3

Linked List Reverse : 3 KMITL 2 1 0

Enter Input : AP 0,AP 1,AP 2,AP 3,SI,IS 999 KMITL,SI

Linked List size = 4 : 0 1 2 3

Linked List size = 5 : 0 1 2 3 KMITL

Linked List : 0 1 2 3 KMITL

Linked List Reverse : KMITL 3 2 1 0

Enter Input : AH 3,AH 2,AH 1,AH 0,SI,IS -3 KMITL,SI

Linked List size = 4 : 0 1 2 3

Linked List size = 5 : 0 KMITL 1 2 3

Linked List : 0 KMITL 1 2 3

Linked List Reverse : 3 2 1 KMITL 0

Enter Input : AP 0,AP 1,AP 2,AP 3,SI,IS -999 KMITL,SI

Linked List size = 4 : 0 1 2 3

Linked List size = 5 : KMITL 0 1 2 3

Linked List : KMITL 0 1 2 3

Linked List Reverse : 3 2 1 0 KMITL

3.

ให้น้องรับ Linked List มา 2 ตัว จากนั้นนำ 2 Linked List มาต่อกัน โดย L2 จะมาต่อ L1 แบบกลับหลัง

Enter Input (L1,L2) : 1 2

L1 : 1

L2 : 2

Merge : 1 2

Enter Input (L1,L2) : 1->2->3 4->5

L1 : 1 2 3

L2 : 4 5

Merge : 1 2 3 5 4

Enter Input (L1,L2) : I->Love->KMITL Datastruct->and

L1 : I Love KMITL

L2 : Datastruct and

Merge : I Love KMITL and Datastruct

Enter Input (L1,L2) : CE->2D Lardkrabang->KMITL

L1 : CE 2D

L2 : Lardkrabang KMITL

Merge : CE 2D KMITL Lardkrabang

ข้อ 4

กฤษฎาได้มีไอเดียสุดบรรเจิดคือการสร้างโปรแกรม Text Editor แบบ VIM ขึ้นมาใช้งานเอง โดยโปรแกรมนี้จะมีอยู่แค่ 1 Mode คือ Command Mode (inputของเรานั่นแหละ) โดยจะมีคำสั่งอยู่ 5 แบบ คือ Insert (I) , Left (L) , Right (R) , Backspace (B) และ Delete (D) (โดยความสามารถของคำสั่งต่างๆจะอธิบายด้านล่าง) แต่กฤษฎาไม่มีความสามารถทางด้านการสร้างโปรแกรมเลย กฤษฎาจึงได้มาขอร้องน้องๆที่เรียนอยู่วิศวกรรมคอมพิวเตอร์ ให้ช่วยสร้างโปรแกรม Text Editor ที่กฤษฎาต้องการให้หน่อย โดยผลลัพธ์ให้แสดงออกมาเป็น word ที่เหลืออยู่จาก Command ที่เราใส่ลงไป พร้อมกับตำแหน่งของ Cursor

\*\*\*\*\* อธิบาย Input 5 แบบ \*\*\*\*\*

1.  I <word > :   เป็นการนำ word ลงไปใส่ในตำแหน่งของ Cursor ปัจจุบัน หลังจากใส่ word เสร็จ ตำแหน่งของ Cursor จะมาอยู่ด้านหลังของ word ที่ใส่ลงไป

2.  L              :   เป็นการเลื่อน Cursor จากตำแหน่งปัจจุบันไปทางด้านซ้าย 1 ตำแหน่ง หากอยู่ทางซ้ายสุดอยู่แล้วจะไม่เกิดผลอะไร

3.  R             :   เป็นการเลื่อน Cursor จากตำแหน่งปัจจุบันไปทางด้านขวา 1 ตำแหน่ง หากอยู่ทางขวาสุดอยู่แล้วจะไม่เกิดผลอะไร

4.  B             :   เป็นการลบ word 1 ตัวทางด้านซ้ายของ Cursor หากอยู่ทางซ้ายสุดอยู่แล้วจะไม่เกิดผลอะไร

5.  D             :   เป็นการลบ word 1 ตัวทางด้านขวาของ Cursor หากอยู่ทางขวาสุดอยู่แล้วจะไม่เกิดผลอะไร

Enter Input : I Apple,I Bird,I Cat

Apple Bird Cat |

Enter Input : I Apple,I Bird,I Cat,L

Apple Bird | Cat

Enter Input : I Apple,I Bird,I Cat,L,L

Apple | Bird Cat

Enter Input : I Apple,I Bird,I Cat,L,L,L,L,L

| Apple Bird Cat

Enter Input : I Apple,I Bird,I Cat,L,L,R

Apple Bird | Cat

Enter Input : I Apple,I Bird,I Cat,L,L,R,B

Apple | Cat

Enter Input : I Apple,I Bird,L,L,R,D,D

Apple |

Enter Input : I I,I KMITL,L,L,R,I Love

I Love | KMITL

Enter Input : I I,I KMITL,L,L,R,I Love,D,I DataStructure,L,L,R,L,R,B,I Hate

I Hate | DataStructure

ให้น้องๆใช้ Linked List (เขียนเป็น class)  ในการทำ Radix Sort  (มีอยู่ในสไลด์เรียน 2 หน้าสุดท้าย)  ในรูปแบบน้อยไปมาก

โดยผลลัพธ์ให้ออกมาเป็นการทำ Radix Sort แบบจำนวนรอบน้อยที่สุด และแสดงผลในแต่ละรอบว่าได้ผลลัพธ์เป็นอย่างไร  3 บรรทัดสุดท้ายจะเป็น ( จำนวนรอบที่น้อยที่สุด , Data ก่อนทำ Radix Sort และ Data หลังทำ Radix Sort )

Enter Input : 64 8 216 512 27 729 0 1 343 125

------------------------------------------------------------

Round : 1

0 : 0

1 : 1

2 : 512

3 : 343

4 : 64

5 : 125

6 : 216

7 : 27

8 : 8

9 : 729

------------------------------------------------------------

Round : 2

0 : 0 1 8

1 : 216 512

2 : 27 125 729

3 :

4 : 343

5 :

6 : 64

7 :

8 :

9 :

------------------------------------------------------------

Round : 3

0 : 0 1 8 27 64

1 : 125

2 : 216

3 : 343

4 :

5 : 512

6 :

7 : 729

8 :

9 :

------------------------------------------------------------

Round : 4

0 : 0 1 8 27 64 125 216 343 512 729

1 :

2 :

3 :

4 :

5 :

6 :

7 :

8 :

9 :

------------------------------------------------------------

3 Time(s)

Before Radix Sort : 64 -> 8 -> 216 -> 512 -> 27 -> 729 -> 0 -> 1 -> 343 -> 125

After Radix Sort : 0 -> 1 -> 8 -> 27 -> 64 -> 125 -> 216 -> 343 -> 512 -> 729

Enter Input : -123 456 -789 0 27 3645 133 -142 -5038594 15615 668 2 -1 72

------------------------------------------------------------

Round : 1

0 : 0

1 : -1

2 : -142 2 72

3 : -123 133

4 : -5038594

5 : 3645 15615

6 : 456

7 : 27

8 : 668

9 : -789

------------------------------------------------------------

Round : 2

0 : -1 0 2

1 : 15615

2 : -123 27

3 : 133

4 : -142 3645

5 : 456

6 : 668

7 : 72

8 : -789

9 : -5038594

------------------------------------------------------------

Round : 3

0 : -1 0 2 27 72

1 : -142 -123 133

2 :

3 :

4 : 456

5 : -5038594

6 : 668 3645 15615

7 : -789

8 :

9 :

------------------------------------------------------------

Round : 4

0 : -789 -142 -123 -1 0 2 27 72 133 456 668

1 :

2 :

3 : 3645

4 :

5 : 15615

6 :

7 :

8 : -5038594

9 :

------------------------------------------------------------

Round : 5

0 : -789 -142 -123 -1 0 2 27 72 133 456 668 3645

1 : 15615

2 :

3 : -5038594

4 :

5 :

6 :

7 :

8 :

9 :

------------------------------------------------------------

Round : 6

0 : -5038594 -789 -142 -123 -1 0 2 27 72 133 456 668 3645 15615

1 :

2 :

3 :

4 :

5 :

6 :

7 :

8 :

9 :

------------------------------------------------------------

5 Time(s)

Before Radix Sort : -123 -> 456 -> -789 -> 0 -> 27 -> 3645 -> 133 -> -142 -> -5038594 -> 15615 -> 668 -> 2 -> -1 -> 72

After Radix Sort : -5038594 -> -789 -> -142 -> -123 -> -1 -> 0 -> 2 -> 27 -> 72 -> 133 -> 456 -> 668 -> 3645 -> 15615

Enter Input : -1 -9 -3 -6 -5 -4 -7 0 -8 -2 3 2 5 1 4 9 8 7 6

------------------------------------------------------------

Round : 1

0 : 0

1 : -1 1

2 : -2 2

3 : -3 3

4 : -4 4

5 : -5 5

6 : -6 6

7 : -7 7

8 : -8 8

9 : -9 9

------------------------------------------------------------

Round : 2

0 : -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9

1 :

2 :

3 :

4 :

5 :

6 :

7 :

8 :

9 :

------------------------------------------------------------

1 Time(s)

Before Radix Sort : -1 -> -9 -> -3 -> -6 -> -5 -> -4 -> -7 -> 0 -> -8 -> -2 -> 3 -> 2 -> 5 -> 1 -> 4 -> 9 -> 8 -> 7 -> 6

After Radix Sort : -9 -> -8 -> -7 -> -6 -> -5 -> -4 -> -3 -> -2 -> -1 -> 0 -> 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9

Enter Input : 15 -15

------------------------------------------------------------

Round : 1

0 :

1 :

2 :

3 :

4 :

5 : -15 15

6 :

7 :

8 :

9 :

------------------------------------------------------------

Round : 2

0 :

1 : -15 15

2 :

3 :

4 :

5 :

6 :

7 :

8 :

9 :

------------------------------------------------------------

Round : 3

0 : -15 15

1 :

2 :

3 :

4 :

5 :

6 :

7 :

8 :

9 :

------------------------------------------------------------

2 Time(s)

Before Radix Sort : 15 -> -15

After Radix Sort : -15 -> 15

Enter Input : 100 0 10

------------------------------------------------------------

Round : 1

0 : 0 10 100

1 :

2 :

3 :

4 :

5 :

6 :

7 :

8 :

9 :

------------------------------------------------------------

0 Time(s)

Before Radix Sort : 100 -> 0 -> 10

After Radix Sort : 0 -> 10 -> 100