ชื่อ – นามสกุล **วัฒชัย เตชะลือ** รหัสนักศึกษา **65015137**

Data Structures and Algorithm
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

การทดลองที่ 9 : Recursive

จุดประสงค์

1. นักศึกษาเข้าใจการทำงานของ Recursive Operation และสามารถออกแบบฟังก์ชันที่ทำงานเป็นแบบ Recursive ตามต้องการได้

ตอนที่ 1 : การออกแบบการทำงานและสร้างฟังก์ชันที่ทำงานแบบ Recursive

1. ให้นักศึกษาเขียนฟังก์ชัน def isPalindrome(str):
   โดย str เป็น string ที่ต้องการตรวจสอบและส่งค่าเป็น true เมื่อ str เป็น palindrome และให้ค่า false เมื่อ str ไม่เป็น palindrome

แนวคิดในการออกแบบการทำงานแบบ recursive

ตรวจสอบความเป็น palindrome ของ string ด้วยการเปรียบเทียบตัวอักษรแรกและตัวอักษรสุดท้ายของ string ถ้าตัวอักษรแรกและตัวอักษรสุดท้ายเหมือนกัน จะลบตัวอักษรเหล่านั้นออกแล้วเรียกฟังก์ชัน

กรณีของ str ที่เป็น base case คืออะไร และ ที่ base case จะมีการทำงานอย่างไร

1. Base case สำหรับฟังก์ชัน isPalindrome คือกรณีที่ str เป็น string ว่างหรือมีความยาวเป็น 1 เพราะ string เหล่านี้ถือว่าเป็น palindrome
2. ในกรณี base case, ฟังก์ชันจะส่งค่า true

กรณีของ str ที่เป็น recursive case จะมีการทำงานอย่างไร

1. Recursive case สำหรับฟังก์ชัน isPalindrome คือกรณีที่ str มีความยาวมากกว่า 1
2. ในกรณี recursive case, จะตรวจสอบว่าตัวอักษรแรกและตัวอักษรสุดท้ายของ str เหมือนกันหรือไม่
   - ถ้าไม่เหมือน, str ไม่เป็น palindrome, จะส่งค่า false
   - ถ้าเหมือน, จะตัดตัวอักษรแรกและตัวอักษรสุดท้ายของ str, เรียกฟังก์ชัน isPalindrome ใหม่ด้วย string ที่ได้

ฟังก์ชันที่ได้

```
def isPalindrome(str):
    if len(str) < 2:
        return True
    elif str[0] != str[-1]:
        return False
    else:
        return isPalindrome(str[1:-1])
```

ชื่อ – นามสกุล **วัฒชัย เตชะลือ** รหัสนักศึกษา **65015137**

ทดสอบการทำงานของฟังก์ชัน

str	ผลลัพธ์
abcdcba	**True**
atoyota	**True**
kmitl	**False**
manassanan	**False**
programming	**False**
fundamental	**False**

3. ให้นักศึกษาเขียนฟังก์ชัน

def isAscending(list_of_integer):

โดย list_of_integer เป็น list ของข้อมูลตัวเลขจำนวนเต็มที่ต้องการตรวจสอบว่าเป็น list ที่เรียงลำดับจากน้อยไปมากแล้วหรือไม่ และส่งค่าเป็น true เมื่อข้อมูลใน list_of_integer เรียงลำดับจากน้อยไปมาก และให้ค่า false เมื่อ list_of_integer ไม่ได้เรียงลำดับจากน้อยไปมาก

แนวคิดในการออกแบบการทำงานแบบ recursive

ตรวจสอบความเรียงลำดับระหว่างค่าตัวแรกและค่าตัวที่สองใน list และหากค่าตัวแรกน้อยกว่าหรือเท่ากับค่าตัวที่สอง จะเรียกฟังก์ชัน isAscending ด้วย list ที่ลดลงและทำเช่นนี้ในทุก iteration จนกว่าจะถึง base case

กรณีของ list ที่เป็น base case คืออะไร และ ที่ base case จะมีการทำงานอย่างไร

1. Base case สำหรับฟังก์ชัน isAscending คือกรณีที่ list_of_integer เป็น list ว่างหรือมีข้อมูลเพียงหนึ่งตัว เพราะ list เหล่านี้ถือว่าเป็น list ที่เรียงลำดับจากน้อยไปมาก
2. ในกรณี base case, ฟังก์ชันจะส่งค่า true

กรณีของ list ที่เป็น recursive case จะมีการทำงานอย่างไร

1. Recursive case สำหรับฟังก์ชัน isAscending คือกรณีที่ list_of_integer มีข้อมูลมากกว่าหนึ่งตัว
2. ในกรณี recursive case, เราจะตรวจสอบว่าตัวเลขตัวแรกของ list_of_integer มีค่าไม่มากกว่าตัวเลขตัวถัดไปหรือไม่
    - ถ้ามีค่ามากกว่า, list_of_integer ไม่ได้เรียงลำดับจากน้อยไปมาก, เราจะส่งค่า false
    - ถ้าไม่มีค่ามากกว่า, เราจะตัดตัวเลขตัวแรกของ list_of_integer, เรียกฟังก์ชัน isAscending ใหม่ด้วย list_of_integer

ชื่อ – นามสกุล **วัฒชัย เตชะลือ** รหัสนักศึกษา **65015137**

ฟังก์ชันที่ได้

```
def isAscending(list_of_integer):
    if len(list_of_integer) < 2:
        return True
    elif list_of_integer[0] > list_of_integer[1]:
        return False
    else:
        return isAscending(list_of_integer[1:])
```

ทดสอบการทำงานของฟังก์ชัน

list_of_integer	ผลลัพธ์
[1,2,3,4,5,6,7]	**True**
[3,4,2,5,6,1,2]	**False**
[9,8,7,6,5,4]	**False**
[0,0,1,1,2,2,3,3,4,4,5,5]	**True**
[6,7,8,9,10,11,12]	**True**
[6,3,8,7,9,2,3,1,5]	**False**

4. ให้นักศึกษาเขียนฟังก์ชัน

def group_of_no_1(island_list,point_no):

โดย

island_list คือ list ของตัวเลข 1 และ 0 ที่เรียงตัวกัน

point_no คือ ตำแหน่งที่ต้องการตรวจสอบ

ฟังก์ชัน group_of_no_1 จะหาจำนวนตัวเลข 1 ที่ตำแหน่ง point_no ว่ามีจำนวนเลข 1 ติดกันกี่ตัว ถ้า island_list = [0,1,1,1,1,0,0,1,1,0,1,0] และ point_no = 2 จะได้ผลลัพธ์คือ 4 เพราะมีตัวเลข 1 เรียงกัน 4 ตัวที่ตำแหน่งนั้น หรือถ้า point_no = 8 จะได้ผลลัพธ์เป็น 2

แนวคิดในการออกแบบการทำงานแบบ recursive

หาจำนวนตัวเลข 1 ที่ตำแหน่ง point_no ได้โดยเริ่มต้นที่ point_no และนับตัวเลข 1 จากตำแหน่งนี้ไปทางซ้ายและขวา โดยเพิ่มเลข 1 ที่เจอในลิสต์เมื่อเราเดินทางไปทางทิศตะวันตก (ซ้าย) และทางทิศตะวันออก (ขวา) หากเราเจอตัวเลข 0 หรือเลขที่อยู่นอกขอบของลิสต์ ให้หยุดการนับและคืนค่าที่นับได้

กรณีของ base case คืออะไร และ ที่ base case จะมีการทำงานอย่างไร

1. Base case สำหรับฟังก์ชัน group_of_no_1 คือกรณีที่ตำแหน่ง point_no เป็นตัวเลข 0 หรือมากกว่าความยาวของ island_list
2. ในกรณี base case, ฟังก์ชันจะส่งค่า 0

ชื่อ – นามสกุล **วัฒชัย เตชะลือ** รหัสนักศึกษา **65015137**

กรณีของ recursive case จะมีการทำงานอย่างไร

1. Recursive case สำหรับฟังก์ชัน group_of_no_1 คือกรณีที่ตำแหน่ง point_no เป็นตัวเลขที่มีค่าไม่เป็น 0
2. ในกรณี recursive case, เราจะตรวจสอบว่าตัวเลขในตำแหน่ง point_no เป็น 1 หรือไม่
   - ถ้าไม่เป็น, เราจะส่งค่า 0
   - ถ้าเป็น, เราจะเพิ่ม point_no ขึ้นไปอีกหนึ่ง, เรียกฟังก์ชัน group_of_no_1 ใหม่ด้วย island_list

ฟังก์ชันที่ได้

```
def group_of_no_1(island_list, point_no):

    count = 0

    left_index = point_no
    while left_index >= 0 and island_list[left_index] == 1:
        count += 1
        left_index -= 1

    right_index = point_no + 1
    while right_index < len(island_list) and island_list[right_index] == 1:
        count += 1
        right_index += 1

    return count
```

ทดสอบการทำงานของฟังก์ชัน

island_list	point_no	ผลลัพธ์
[1,1,1,1,0,0,0,1,1,1,0,0]	1	**4**
[1,1,1,1,0,0,0,1,1,1,0,0]	5	**0**
[1,0,1,1,1,0,0,0,1,1,1,1,1,1]	4	**3**
[1,0,1,1,1,0,0,0,1,1,1,1,1,1]	10	**6**
[1,0,1,1,1,0,0,0,1,1,1,1,1,1]	1	**3**
[0,1,0,1,0,1,0,1,0,1]	7	**1**

5. ให้นักศึกษาเขียนฟังก์ชัน

   def valid_parentheses(str):

   โดย

   str คือ string ของวงเล็บในรูปแบบต่างๆ

   ฟังก์ชั่น valid_parentheses จะตรวจสอบว่าวงเล็บใน str มีการจัดลำดับได้อย่างถูกต้องหรือไม่

   ถ้า str = "(()()(()))"จะได้ผลลัพธ์คือ True , ถ้า str = "(((()()()(())))"จะได้ผลลัพธ์คือ False

แนวคิดในการออกแบบการทำงานแบบ recursive

ชื่อ – นามสกุล **วัฒชัย เตชะลือ** รหัสนักศึกษา **65015137**

ตรวจสอบว่าวงเล็บที่เราเจอในสตริง str มีการจัดลำดับถูกต้องหรือไม่ โดยเริ่มต้นจากวงเล็บเปิด "(" และหาวงเล็บปิด ")" ในกรณีที่เราไม่พบวงเล็บปิดที่สอดคล้องกันเราจะสามารถคืนค่า False

กรณีของ base case คืออะไร และ ที่ base case จะมีการทำงานอย่างไร

Base case คือกรณีที่ไม่มีวงเล็บใดๆ ที่เหลือในสตริง str ที่เราต้องตรวจสอบ. ในกรณีนี้ ถ้า str เป็นสตริงว่าง คือ str == "" และไม่มีวงเล็บที่เราต้องตรวจสอบอีก และในกรณีนี้เราจะคืนค่า True เพราะวงเล็บทั้งหมดใน str ถูกจัดลำดับถูกต้อง.

กรณีของ recursive case จะมีการทำงานอย่างไร

ในกรณี recursive เราจะหาวงเล็บเปิดแรกที่เราพบใน str และหาวงเล็บปิดที่สอดคล้องกัน หากเราพบวงเล็บเปิดก่อนวงเล็บปิดเราจะเรียกฟังก์ชัน valid_parentheses ด้วยสตริงที่อยู่ระหว่างวงเล็บเปิดและวงเล็บปิดนั้น และตรวจสอบว่าสตริงย่อยนี้ถูกต้องหรือไม่

ฟังก์ชันที่ได้

```
def valid_parentheses(str):
    # Base case
    if str == "":
        return True

    open_paren = str.find("(")
    if open_paren == -1:
        return False

    close_paren = str.find(")", open_paren)
    if close_paren == -1:
        return False

    return valid_parentheses(str[:open_paren] + str[open_paren + 1:close_paren] + str[close_paren + 1:])
```

ทดสอบการทำงานของฟังก์ชัน

str	ผลลัพธ์
(()()(())())	**True**
((()()	**False**
())()()(	**False**
((((()))(((())))	**True**
()()(((())))	**True**
()	**True**