**Recursion**

**1.** \*\*\*\*\*\* ห้ามใช้ For , While  ( ให้ฝึกเอาไว้ เนื่องจากถ้าเจอตอนสอบจะได้ 0 )

ให้เขียน Recursive หาค่า Max ของ Input

**Enter Input : 8 7 10 1 5 4 2 6 3 9**

**Max : 10**

**Enter Input : -84 -230 -54845 -6 -1**

**Max : -1**

**2.** \*\*\*\*\*\* ห้ามใช้ For , While  ( ให้ฝึกเอาไว้ เนื่องจากถ้าเจอตอนสอบจะได้ 0 )

เขียน Recursive เพื่อหาว่า String ที่รับเข้ามาเป็น Palindrome หรือไม่

**Enter Input : abba**

**'abba' is palindrome**

**Enter Input : abgba**

**'abgba' is palindrome**

**Enter Input : abcdefkgfedfe**

**'abcdefkgfedfe' is not palindrome**

**3.**เขียนโปรแกรมสำหรับหา หรม. ของเลข 2 ตัว

**\*\*\*\*ห้ามใช้คำสั่ง len, for, while, do while หรือ \*\*\*\*\***

**หมายเหตุ ฟังก์ชันต้องมี parameter แค่เพียง 2 ตัว**

**บทนิยาม**

**ตัวหารร่วมมาก** หรือ **ห.ร.ม.** ([อังกฤษ](https://th.wikipedia.org/wiki/%E0%B8%A0%E0%B8%B2%E0%B8%A9%E0%B8%B2%E0%B8%AD%E0%B8%B1%E0%B8%87%E0%B8%81%E0%B8%A4%E0%B8%A9): greatest common divisor: gcd) ของ[จำนวนเต็ม](https://th.wikipedia.org/wiki/%E0%B8%88%E0%B8%B3%E0%B8%99%E0%B8%A7%E0%B8%99%E0%B9%80%E0%B8%95%E0%B9%87%E0%B8%A1)สองจำนวนซึ่งไม่เป็นศูนย์พร้อมกัน คือจำนวนเต็มที่มากที่สุดที่[หาร](https://th.wikipedia.org/wiki/%E0%B8%AB%E0%B8%B2%E0%B8%A3)ทั้งสองจำนวนลงตัว

**Enter Input : 8 4**

**The gcd of 8 and 4 is : 4**

**Enter Input : 10 20**

**The gcd of 20 and 10 is : 10**

**Enter Input : 12 18**

**The gcd of 18 and 12 is : 6**

**Enter Input : 9 7**

**The gcd of 9 and 7 is : 1**

**Enter Input : 0 5**

**The gcd of 5 and 0 is : 5**

**Enter Input : -6 9**

**The gcd of 9 and -6 is : 3**

**Enter Input : -24 -36**

**The gcd of -36 and -24 is : 12**

**Enter Input : 0 0**

**Error! must be not all zero.**

**4.**เขียนโปรแกรมแก้ปัญหา **หอคอยแห่งฮานอย**โดยเราจะมีแทงไม้อยู่3แท่งคือ A B C และรับ input เป็นจำนวนแผ่นไม้ที่วางซ้อนกันให้แสดงลำดับการย้ายแผ่นไม้ทั้งหมดจากแท่ง A ไปยัง แท่งC โดยแผ่นไม้ที่มีขนาดเล็กกว่าจะอยู่ข้างบนแผ่นไม้ที่มีขนาดใหม่กว่าเสมอ(ห้ามวางแผ่นเล็กกว่าไว้ข้างล่าง)

**\*\*\*\*ห้ามใช้คำสั่ง for, while, do while\*\*\*\*\***

**หมายเหตุ ทุกฟังก์ชันต้องมี parameter มากที่สุดไม่เกิน 5 ตัว**

**คำแนะนำ ให้สร้างฟังก์ชันสำหรับแสดงผล แยกต่างหาก และใช้ list ในการเก็บข้อมูลของแท่งไม้แต่ละแท่ง  
และให้ระวังเรื่องการสลับ list ให้ดีๆ**

หากมีข้อสงสัยเกี่ยวกับ **หอคอยแห่งฮานอย**สามารถสอบถาม TA เพิ่มเติม หรือ ลองเล่นได้ที่ <https://www.mathsisfun.com/games/towerofhanoi.html>

def move(n,A,B,C,maxn):

    #code here

n = int(input("Enter Input : "))

**Enter Input : 3**

**| | |**

**1 | |**

**2 | |**

**3 | |**

**move 1 from A to C**

**| | |**

**| | |**

**2 | |**

**3 | 1**

**move 2 from A to B**

**| | |**

**| | |**

**| | |**

**3 2 1**

**move 1 from C to B**

**| | |**

**| | |**

**| 1 |**

**3 2 |**

**move 3 from A to C**

**| | |**

**| | |**

**| 1 |**

**| 2 3**

**move 1 from B to A**

**| | |**

**| | |**

**| | |**

**1 2 3**

**move 2 from B to C**

**| | |**

**| | |**

**| | 2**

**1 | 3**

**move 1 from A to C**

**| | |**

**| | 1**

**| | 2**

**| | 3**

**Enter Input : 4**

**| | |**

**1 | |**

**2 | |**

**3 | |**

**4 | |**

**move 1 from A to B**

**| | |**

**| | |**

**2 | |**

**3 | |**

**4 1 |**

**move 2 from A to C**

**| | |**

**| | |**

**| | |**

**3 | |**

**4 1 2**

**move 1 from B to C**

**| | |**

**| | |**

**| | |**

**3 | 1**

**4 | 2**

**move 3 from A to B**

**| | |**

**| | |**

**| | |**

**| | 1**

**4 3 2**

**move 1 from C to A**

**| | |**

**| | |**

**| | |**

**1 | |**

**4 3 2**

**move 2 from C to B**

**| | |**

**| | |**

**| | |**

**1 2 |**

**4 3 |**

**move 1 from A to B**

**| | |**

**| | |**

**| 1 |**

**| 2 |**

**4 3 |**

**move 4 from A to C**

**| | |**

**| | |**

**| 1 |**

**| 2 |**

**| 3 4**

**move 1 from B to C**

**| | |**

**| | |**

**| | |**

**| 2 1**

**| 3 4**

**move 2 from B to A**

**| | |**

**| | |**

**| | |**

**| | 1**

**2 3 4**

**move 1 from C to A**

**| | |**

**| | |**

**| | |**

**1 | |**

**2 3 4**

**move 3 from B to C**

**| | |**

**| | |**

**| | |**

**1 | 3**

**2 | 4**

**move 1 from A to B**

**| | |**

**| | |**

**| | |**

**| | 3**

**2 1 4**

**move 2 from A to C**

**| | |**

**| | |**

**| | 2**

**| | 3**

**| 1 4**

**move 1 from B to C**

**| | |**

**| | 1**

**| | 2**

**| | 3**

**| | 4**

**5.**   
เขียนโปรแกรมที่แสดงผลดังตัวอย่าง

**\*\*\*\*ห้ามใช้คำสั่ง for, while, do while\*\*\*\*\***

**หมายเหตุ ฟังก์ชันมี parameter ได้ไม่เกิน 2 ตัว**

def staircase(n):

    #code here

print(staircase(int(input("Enter Input : "))))

**Enter Input : 3**

**\_\_#**

**\_##**

**Enter Input : 7**

**\_\_\_\_\_\_#**

**\_\_\_\_\_##**

**\_\_\_\_###**

**\_\_\_####**

**\_\_#####**

**\_######**

**Enter Input : 0**

**Not Draw!**

**Tree1**

**1.**ให้น้องรับ input แล้วนำ input นั้นมาสร้าง Binary Search Tree โดย input ตัวแรกสุดจะเป็น Root เสมอ

class Node:

    def \_\_init\_\_(self, data):

        self.data = data

        self.left = None

        self.right = None

    def \_\_str\_\_(self):

        return str(self.data)

class BST:

    def \_\_init\_\_(self):

        self.root = None

    def insert(self, data):

        # Code Here

    def printTree(self, node, level = 0):

        if node != None:

            self.printTree(node.right, level + 1)

            print('     ' \* level, node)

            self.printTree(node.left, level + 1)

T = BST()

inp = [int(i) for i in input('Enter Input : ').split()]

for i in inp:

    root = T.insert(i)

T.printTree(root)

**Enter Input : 10 4 20 1 5**

**20**

**10**

**5**

**4**

**1**

**Enter Input : 4 10 3 6 13 9**

**13**

**10**

**9**

**6**

**4**

**3**

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

**8**

**7**

**6**

**5**

**4**

**3**

**2**

**1**

**0**

**-1**

**-2**

**2.**ให้น้องรับ input แล้วนำ input นั้นมาสร้าง Binary Search Tree โดย input ตัวแรกสุดจะเป็น Root เสมอ และหาค่าที่น้อยและมากที่สุดของ Binary Search Tree

\*\*\*\*\* ห้ามใช้ Built-in Function เช่น min() , max() , sort() , sorted()

**Enter Input : 10 4 20 1 5**

**20**

**10**

**5**

**4**

**1**

**--------------------------------------------------**

**Min : 1**

**Max : 20**

**Enter Input : 4 10 3 6 13 9**

**13**

**10**

**9**

**6**

**4**

**3**

**--------------------------------------------------**

**Min : 3**

**Max : 13**

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

**9**

**8**

**7**

**6**

**5**

**4**

**3**

**2**

**1**

**0**

**-1**

**-2**

**--------------------------------------------------**

**Min : -2**

**Max : 9**

**3.**ให้น้องๆรับ input เป็น list และ k โดยให้สร้าง Binary Search Tree จาก list ที่รับมา และหลังจากนั้นให้ทำการดูว่าใน Tree มีค่าไหนที่มากกว่าค่า k หรือไม่ ถ้ามีให้ทำการคูณ 3 เพิ่มเข้าไป

**Enter Input : 67 102 81 35 15 7 99 196 202 152/90**

**202**

**196**

**152**

**102**

**99**

**81**

**67**

**35**

**15**

**7**

**--------------------------------------------------**

**606**

**588**

**456**

**306**

**297**

**81**

**67**

**35**

**15**

**7**

**Enter Input : 5 3 -1 4 7 6 8/-5**

**8**

**7**

**6**

**5**

**4**

**3**

**-1**

**--------------------------------------------------**

**24**

**21**

**18**

**15**

**12**

**9**

**-3**

**Enter Input : 5 3 1 4 7 6 8/4**

**8**

**7**

**6**

**5**

**4**

**3**

**1**

**--------------------------------------------------**

**24**

**21**

**18**

**15**

**4**

**3**

**1**

**4**ให้น้องรับ input เข้ามาและสร้าง Binary Search Tree ต่อมาให้แสดงผลแบบ Preorder , Inorder , Postorder และ Breadth First Search ตามลำดับ

**Enter Input : 10 4 20 1 5**

**Preorder : 10 4 1 5 20**

**Inorder : 1 4 5 10 20**

**Postorder : 1 5 4 20 10**

**Breadth : 10 4 20 1 5**

**Enter Input : 0 -50 50 25 -25 13 -13 28 -38 75 -75 62 -62 100 -100**

**Preorder : 0 -50 -75 -100 -62 -25 -38 -13 50 25 13 28 75 62 100**

**Inorder : -100 -75 -62 -50 -38 -25 -13 0 13 25 28 50 62 75 100**

**Postorder : -100 -62 -75 -38 -13 -25 -50 13 28 25 62 100 75 50 0**

**Breadth : 0 -50 50 -75 -25 25 75 -100 -62 -38 -13 13 28 62 100**

**5.**ให้น้องๆรับ input เป็น postfix จากนั้นให้แปลงเป็น Expression Tree , Infix และ Prefix  โดย Operator จะมีแค่ + - \* /

**Enter Postfix : ab+cde+\*\***

**Tree :**

**e**

**+**

**d**

**\***

**c**

**\***

**b**

**+**

**a**

**--------------------------------------------------**

**Infix : ((a+b)\*(c\*(d+e)))**

**Prefix : \*+ab\*c+de**

**Enter Postfix : abc\*+de\*f+g\*+**

**Tree :**

**g**

**\***

**f**

**+**

**e**

**\***

**d**

**+**

**c**

**\***

**b**

**+**

**a**

**--------------------------------------------------**

**Infix : ((a+(b\*c))+(((d\*e)+f)\*g))**

**Prefix : ++a\*bc\*+\*defg**

**Enter Postfix : ab+c\*de-fg+\*-**

**Tree :**

**g**

**+**

**f**

**\***

**e**

**-**

**d**

**-**

**c**

**\***

**b**

**+**

**a**

**--------------------------------------------------**

**Infix : (((a+b)\*c)-((d-e)\*(f+g)))**

**Prefix : -\*+abc\*-de+fg**

**Tree 2**

**1.** กฤษฎาได้ค้นพบเม็ดถั่ววิเศษที่เมื่อโยนลงดินแล้วถั่วจะสามารถเติบโตขึ้นและกลายเป็น Binary Search Tree (BST) ได้ โดยงานของนักศึกษาก็คือนักศึกษาจะต้องสร้าง BST ตามลำดับของข้อมูลนำเข้าซึ่งเป็นตัวเลขจำนวนเต็มที่ไม่ซ้ำกันเลย โดยในการใส่ค่าในแต่ละครั้งจะกลับมาที่ Root of BST เสมอ  แล้วท่องต้นไม้ไปทางซ้ายด้วยคำสั่ง "L" หรือท่องต้นไม้ไปทางขวาด้วยคำสั่ง "R" จนกว่าจะถึงตำแหน่งที่เหมาะสมที่จะใส่ข้อมูลแล้วจึงพิมพ์ "\*" เพื่อใส่ข้อมูลลงไปในต้นไม้  จงเขียนโปรแกรมเพื่อแสดงคำสั่งการท่องต้นไม้ในการใส่ข้อมูลทีละค่าตามลำดับของข้อมูลนำเข้า

**Enter Input : 1 2 5 4 3 -2 -1**

**\***

**R\***

**RR\***

**RRL\***

**RRLL\***

**L\***

**LR\***

**Enter Input : 1 2 5 4 3 -2 -1**

**\***

**R\***

**RR\***

**RRL\***

**RRLL\***

**L\***

**LR\***

**2.**ให้น้องๆสร้าง AVL Tree ด้วย Class โดยผลลัพธ์ให้แสดงเป็น Tree ในแต่ละรอบหลังจาก Insert และปรับ Balance เรียบร้อยแล้ว

\*\* ถ้าสงสัยสามารถดู visualization ของ AVL ได้ที่ website นี้ : <https://www.cs.usfca.edu/~galles/visualization/AVLtree.html>

class Node:

    def \_\_init\_\_(self, data):

        self.data = data

        self.left = None

        self.right = None

    def \_\_str\_\_(self):

        return str(self.data)

class AVL:

    ?????

    def printTree(self, node, level = 0):

        if node != None:

            self.printTree(node.right, level + 1)

            print('     ' \* level, node)

            self.printTree(node.left, level + 1)

    ?????

Top of Form

You have got full mark !!!

Bottom of Form

**Enter Input : 50 40 35 30 20 10 5**

**Insert : ( 50 )**

**50**

**--------------------------------------------------**

**Insert : ( 40 )**

**50**

**40**

**--------------------------------------------------**

**Insert : ( 35 )**

**50**

**40**

**35**

**--------------------------------------------------**

**Insert : ( 30 )**

**50**

**40**

**35**

**30**

**--------------------------------------------------**

**Insert : ( 20 )**

**50**

**40**

**35**

**30**

**20**

**--------------------------------------------------**

**Insert : ( 10 )**

**50**

**40**

**35**

**30**

**20**

**10**

**--------------------------------------------------**

**Insert : ( 5 )**

**50**

**40**

**35**

**30**

**20**

**10**

**5**

**Enter Input : 50 40 35 30 20 10 5**

**Insert : ( 50 )**

**50**

**--------------------------------------------------**

**Insert : ( 40 )**

**50**

**40**

**--------------------------------------------------**

**Insert : ( 35 )**

**50**

**40**

**35**

**--------------------------------------------------**

**Insert : ( 30 )**

**50**

**40**

**35**

**30**

**--------------------------------------------------**

**Insert : ( 20 )**

**50**

**40**

**35**

**30**

**20**

**--------------------------------------------------**

**Insert : ( 10 )**

**50**

**40**

**35**

**30**

**20**

**10**

**--------------------------------------------------**

**Insert : ( 5 )**

**50**

**40**

**35**

**30**

**20**

**10**

**5**

**Enter Input : 40 20 10 25 30 22 50**

**Insert : ( 40 )**

**40**

**--------------------------------------------------**

**Insert : ( 20 )**

**40**

**20**

**--------------------------------------------------**

**Insert : ( 10 )**

**40**

**20**

**10**

**--------------------------------------------------**

**Insert : ( 25 )**

**40**

**25**

**20**

**10**

**--------------------------------------------------**

**Insert : ( 30 )**

**40**

**30**

**25**

**20**

**10**

**--------------------------------------------------**

**Insert : ( 22 )**

**40**

**30**

**25**

**22**

**20**

**10**

**--------------------------------------------------**

**Insert : ( 50 )**

**50**

**40**

**30**

**25**

**22**

**20**

**10**

**Enter Input : 30 40 10 50 20 5 35**

**Insert : ( 30 )**

**30**

**--------------------------------------------------**

**Insert : ( 40 )**

**40**

**30**

**--------------------------------------------------**

**Insert : ( 10 )**

**40**

**30**

**10**

**--------------------------------------------------**

**Insert : ( 50 )**

**50**

**40**

**30**

**10**

**--------------------------------------------------**

**Insert : ( 20 )**

**50**

**40**

**30**

**20**

**10**

**--------------------------------------------------**

**Insert : ( 5 )**

**50**

**40**

**30**

**20**

**10**

**5**

**--------------------------------------------------**

**Insert : ( 35 )**

**50**

**40**

**35**

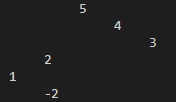
**30**

**20**

**10**

**5**

**3.**จงเขียนฟังก์ชั่นในการหา Rank ของ input ที่รับเข้ามา โดย Rank คือการแบ่งเป็นชั้นๆตามข้อมูลของ BST โดยจะเริ่มจากค่าที่น้อยกว่าค่าใน BST ที่น้อยที่สุดจะมีค่า Rank = 0 และค่าที่อยู่ตั้งแต่ค่าที่น้อยที่สุดจนถึงตัวถัดไปจะมีค่า Rank +=1 ไปเรื่อยๆจนถึงชั้นของตัวสุดท้ายหรือตัวมากสุด เช่น

****

จากรูป ค่าที่น้อยที่สุดคือ -2 ดังนั้น rank(-2) จะได้ 1 แต่ rank ของค่าที่น้อยกว่า -2 จะเท่ากับ 0

และ rank(0) จะเท่ากับ 1 ส่วน rank(1) จะเท่ากับ 2 เป็นต้น

**Enter Input : 1 2 5 4 3 -2/4**

**5**

**4**

**3**

**2**

**1**

**-2**

**--------------------------------------------------**

**Rank of 4 : 5**

**Enter Input : 7 4 3 1 2 6 9 12 5 11/10**

**12**

**11**

**9**

**7**

**6**

**5**

**4**

**3**

**2**

**1**

**--------------------------------------------------**

**Rank of 10 : 8**

**Enter Input : 7 4 2 1 9 8 11/5**

**11**

**9**

**8**

**7**

**4**

**2**

**1**

**--------------------------------------------------**

**Rank of 5 : 3**

**Enter Input : 1 2 4 9 8 5 3/10**

**9**

**8**

**5**

**4**

**3**

**2**

**1**

**--------------------------------------------------**

**Rank of 10 : 7**

**4.** Jean รักษาการผู้บัญชาการของกองอัศวิน Favonius แห่ง Mondstadt ต้องการทราบถึงขุมพลังของอัศวินในแต่ละกลุ่มภายในเมือง Mondstadt แห่งนี้จึงจะทดสอบความแข็งแกร่งของขุมกำลังที่มี โดยจะทำการจัดวางกำลังอัศวินภายในเมือง Mondstadt ดังตัวอย่างต่อไปนี้

                พลัง    :   5  4  4  3  2  2  2

                ลำดับ  :   0  1  2  3  4  5  6

จากข้อมูลข้างต้นประกอบด้วยอัศวินทั้งหมด 7 คน เรียงตามลำดับตั้งแต่ลำดับที่ 0 ถึง 6 และพลังของอัศวินแต่ละคนมีข้อกำหนดดังนี้

    -  อัศวินลำดับที่ n จะมีลูกน้องในสังกัดอยู่ลำดับที่ 2n+1 และ 2n+2 (ลูกน้องของลูกน้องของอัศวินลำดับที่ n ถือว่าเป็นลูกน้องของอัศวินลำดับที่ n ด้วย)

    -  ค่าพลังของอัศวินมีค่าตั้งแต่ 0 - 5

    -  กลุ่มของอัศวินกลุ่มที่ i จะมีสมาชิกคือ อัศวินลำดับที่ i และลูกน้องของอัศวินลำดับที่ i (รวมลูกน้องของลูกน้องของอัศวินด้วย)

    -  พลังของกลุ่มอัศวินลำดับที่ i เป็นพลังรวมของสมาชิกของอัศวินทั้งหมดในกลุ่ม เช่น

            -  อัศวินกลุ่มที่ 1 หมายถึง กลุ่มของอัศวินลำดับที่ 1 ซึ่งมีสมาชิกประกอบด้วย อัศวินลำดับที่ 1, 3 และ 4 และค่าพลังรวมของอัศวินกลุ่มที่ 1 เท่ากับ 4 + 3 + 2 = 9

            -  อัศวินกลุ่มที่ 2 หมายถึง กลุ่มของอัศวินลำดับที่ 2 ซึ่งมีสมาชิกประกอบด้วย อัศวินลำดับที่ 2 , 5 และ 6 และค่าพลังรวมของอัศวินกลุ่มที่ 2 เท่ากับ 4 + 2 + 2 = 8

ดังนั้นเมื่อนำพลังของอัศวินกลุ่มที่ 1 และ 2 มาเทียบกัน จะได้ว่าพลังรวมของอัศวินกลุ่มที่ 1 นั้นมากกว่าพลังรวมของอัศวินกลุ่มที่ 2

Jean ต้องการทราบว่าค่าพลังรวมของอัศวินภายในเมือง Mondstadt เป็นเท่าใด และถ้าเปรียบเทียบระหว่างอัศวินแต่ละกลุ่มแล้วค่าของพลังรวมของอัศวินในกลุ่มใดมีค่ามากกว่ากัน

**Enter Input : 5 4 4 3 2 2 2/1 2,5 6,2 0**

**22**

**1>2**

**5=6**

**2<0**

**Enter Input : 4 5/0 1,1 0**

**9**

**0>1**

**1<0**

**5.**จงเขียนฟังก์ชัน สำหรับตรวจสอบว่า Tree นี้เป็น binary search tree หรือไม่  
โดยกำหนดให้ 0<=node.data<=100 และฟังก์ชั่นมีparameterมากที่สุด4ตัว

class Node:

    def \_\_init\_\_(self, data):

        self.data = data

        self.left = None

        self.right = None

        self.level = None

    def \_\_str\_\_(self):

        return str(self.data)

class Tree:

    def \_\_init\_\_(self):

        self.root = None

        self.num = 0

    def insert(self, val):

        if self.root == None:

            self.root = Node(val)

            self.num += 1

        else:

            h = height(self.root)

            max\_node = pow(2,h+1)-1

            current = self.root

            if self.num+1 > max\_node:

                while(current.left != None):

                    current = current.left

                current.left = Node(val)

                self.num+=1

            elif self.num+1 == max\_node:

                while(current.right != None):

                    current = current.right

                current.right = Node(val)

                self.num+=1

            else:

                print(max\_node-((max\_node-(pow(2,h)-1))/2))

                if self.num+1 <= max\_node-((max\_node-(pow(2,h)-1))/2):

                    insert\_subtree(current.left,self.num - round(pow(2,h)/2),val)

                else:

                    insert\_subtree(current.right,self.num - pow(2,h),val)

                self.num+=1

def insert\_subtree(r,num,val):

    if r != None:

        h = height(r)

        max\_node = pow(2,h+1)-1

        current = r

        if num+1 > max\_node:

            while(current.left != None):

                current = current.left

            current.left = Node(val)

            return

        elif num+1 == max\_node:

            while(current.right != None):

                current = current.right

            current.right = Node(val)

            return

        if num+1 <= max\_node-((max\_node-(pow(2,h)-1))/2):

            insert\_subtree(current.left,num - round(pow(2,h)/2),val)

        else:

            insert\_subtree(current.right,num - pow(2,h),val)

    else:

        return

def height(root):

    if root == None:

        return -1

    else:

        left = height(root.left)

        right = height(root.right)

        if left>right:

            return left + 1

        else:

            return right + 1

def printTree90(node, level = 0):

    if node != None:

        printTree90(node.right, level + 1)

        print('     ' \* level, node)

        printTree90(node.left, level + 1)

def check\_binary\_search\_tree\_(root):

    #code here

tree = Tree()

data = input("Enter Input : ").split()

for e in data:

    tree.insert(int(e))

printTree90(tree.root)

print(check\_binary\_search\_tree\_(tree.root))

**Enter Input : 2 1 3**

**3**

**2**

**1**

**True**

**Enter Input : 1 2 3**

**3**

**1**

**2**

**False**

**Enter Input : 1 0 5 -1**

**5**

**1**

**0**

**-1**

**False**