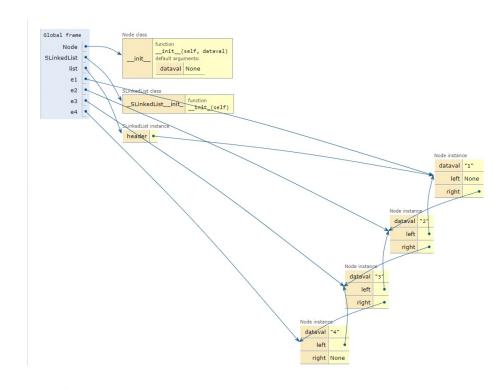
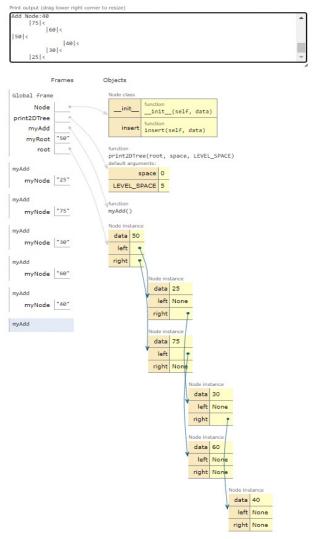
```
class Node:
   2
          def __init__(self, dataval=None):
              self.dataval = dataval
              self.left = None
   4
   5
              self.right = None
   6
      class SLinkedList:
          def __init_(self):
   8
              self.header = None
   9
      list = SLinkedList()
      e1 = Node("1")
  10
      e2 = Node("2")
  11
  12
      e3 = Node("3")
  13
      e4 = Node("4")
  14
      list.header = e1
      e1.right = e2
  16
      e2.left = e1
  17
      e2.right = e3
  18
      e3.left = e2
      e3.right = e4
→ 20
      e4.left = e3
```

(KIIOWII IIIIIICACIONIS)

```
class Node:
    def __init__(self,data):
         self.left = None
self.right = None
         self.data = data
     def insert(self,data):
         if self.data is None:
             self.data = data
             if data < self.data:
    if self.left is None:
        self.left = Node(data)</pre>
                      print2DTree(root)
                  else:
                      self.left.insert(data)
             elif data > self.data:
                  if self.right is None:
    self.right = Node(data)
                      print2DTree(root)
                  else:
                      self.right.insert(data)
def print2DTree(root, space=0, LEVEL_SPACE=5):
    if (root == None):
         return
     space += LEVEL_SPACE
     print2DTree(root.right, space)
     # print() # neighbor space
    print("|" + str(root.data) + "|<")</pre>
    print2DTree(root.left, space)
def myAdd():
         myNode = input('Add Node:')
         if myNode.isdigit():
             root.insert(int(myNode))
             myAdd()
         else:
             print('Only INT!')
while (1):
    myRoot = input('Add Root:')
     if myRoot.isdigit():
         root = Node(int(myRoot))
         print2DTree(root)
         myAdd()
     else:
        print('Only INT!')
```





```
class Node:
                                                def print2DTree(root, space=0,
             _(self, data):
  def init
                                                LEVEL_SPACE=5):
    self.left = None
                                                  if (root == None):
     self.right = None
                                                     return
     self.data = data
                                                  space += LEVEL_SPACE
                                                  print2DTree(root.right, space)
  def insert(self, data):
                                                   # print() # neighbor space
     if self.data is None:
                                                  for i in range(LEVEL_SPACE, space):
       self.data = data
                                                     print(end=" ")
     else:
                                                   print("I" + str(root.data) + "I<")
       if data < self.data:
                                                  print2DTree(root.left, space)
          if self.left is None:
             self.left = Node(data)
            print2DTree(root)
                                               def myDel(root):
          else:
                                                  while (1):
            self.left.insert(data)
                                                     delTarget = int(input('Del Node: '))
       elif data > self.data:
                                                     root = delNode(root, delTarget)
          if self.right is None:
                                                     print2DTree(root)
            self.right = Node(data)
            print2DTree(root)
                                                def myAdd():
            self.right.insert(data)
                                                   while (1):
                                                     myNode = input('Add Node:')
                                                     if myNode.isdigit():
def lastNodeDel(root, myLastNode):
                                                       root.insert(int(myNode))
  myQueue = []
                                                       myAdd()
  myQueue.append(root)
                                                     else:
  while (len(myQueue)):
                                                       myDel(root)
     temp = myQueue.pop(0)
     if temp is myLastNode:
       temp = None
                                                while (1):
       return
                                                   myRoot = input('Add Root:')
     if temp.right:
                                                   if myRoot.isdigit():
       if temp.right is myLastNode:
                                                     root = Node(int(myRoot))
          temp.right = None
                                                     print2DTree(root)
          return
                                                     myAdd()
       else:
                                                   else:
         myQueue.append(temp.right)
                                                     print('Only INT!')
    if temp.left:
       if temp.left is myLastNode:
          temp.left = None
          return
       else
          myQueue.append(temp.left)
def delNode(root, delTarget):
  if root == None:
    return None
  if root.left is None and root.right is None:
     if root.data == delTarget:
       temp = root.right
       root = None
       return temp
  delTarget_node = None
  myQueue = []
  myQueue.append(root)
  temp = None
  while (len(myQueue)):
     temp = myQueue.pop(0)
     if temp.data == delTarget:
       delTarget_node = temp
     if temp.left:
       myQueue.append(temp.left)
     if temp.right:
       myQueue.append(temp.right)
  if delTarget_node:
```

lastNodeVal = temp.data lastNodeDel(root, temp)

return root

delTarget_node.data = lastNodeVal

```
Del Node: 30
      75 <
           |60|<
|50|<
           |40|<
     25 <
Del Node: 75
     60 <
|50|<
           40 <
     25 <
Del Node: 40
     60 <
50 <
Del Node:
```

```
import collections
                                                                                         def lastNodeDel(root, myLastNode):
                                                                                           myQueue = []
class Node:
                                                                                           myQueue.append(root)
                                                                                           while (len(myQueue)):
  def __init__(self, data):
     self.left = None
                                                                                              temp = myQueue.pop(0)
                                                                                              if temp is myLastNode:
     self.right = None
                                                                                                 temp = None
     self.data = data
                                                                                                return
                                                                                              if temp.right:
  def insert(self, data):
                                                                                                if temp.right is myLastNode:
     if self.data is None:
                                                                                                   temp.right = None
        self.data = data
                                                                                                   return
                                                                                                else:
        if data < self.data:
                                                                                                   myQueue.append(temp.right)
           if self.left is None:
                                                                                              if temp.left:
              self.left = Node(data)
                                                                                                 if temp.left is myLastNode:
              print2DTree(root)
                                                                                                   temp.left = None
             print("Height of tree is: " + str(height(root)))
                                                                                                   return
           else:
                                                                                                   myQueue.append(temp.left)
             self.left.insert(data)
        elif data > self.data:
           if self.right is None:
                                                                                        def delNode(root, delTarget):
  if root == None:
             self.right = Node(data)
             print2DTree(root)
                                                                                              return None
              print("Height of tree is: " + str(height(root)))
                                                                                           if root.left is None and root.right is None:
                                                                                              if root.data == delTarget:
             self.right.insert(data)
                                                                                                 temp = root.right
                                                                                                 root = None
def child(obj, data):
                                                                                                return temp
  if obj:
                                                                                           delTarget_node = None
     if obj.data == data:
                                                                                           myQueue = []
        print('Child of node '+' '+str(data)+' is '+str(obj.left.data) +
                                                                                           myQueue.append(root)
                                                                                           temp = None
             ' and '+str(obj.right.data))
                                                                                           while (len(myQueue)):
        return
                                                                                              temp = myQueue.pop(0)
     else:
                                                                                              if temp.data == delTarget:
        return child(obj.left, data) or child(obj.right, data) # เชควน
                                                                                                delTarget_node = temp
                                                                                              if temp.left:
                                                                                                myQueue.append(temp.left)
def parent(obj, data):
                                                                                              if temp.right:
  if obj:
                                                                                                 myQueue.append(temp.right)
     if (obj.left and obj.left.data == data) or (obj.right and obj.right.data == data): if delTarget_node:
        return obj.data or parent(obj.left, data) or parent(obj.right, data)
                                                                                              lastNodeVal = temp data
                                                                                              lastNodeDel(root, temp)
                                                                                              delTarget_node.data = lastNodeVal
        return parent(obj.left, data) or parent(obj.right, data)
                                                                                           return root
def sibling(obj, data):
                                                                                         def print2DTree(root, space=0, LEVEL_SPACE=5):
  if obj:
                                                                                           if (root == None):
     if (obj.left and obj.left.data == data):
                                                                                              return
        return obj.right.data
                                                                                           space += LEVEL_SPACE
print2DTree(root.right, space)
     elif (obj.right and obj.right.data == data):
        return obj.left.data
                                                                                           # print() # neighbor space
for i in range(LEVEL_SPACE, space):
     else:
        return sibling(obj.left, data) or sibling(obj.right, data)
                                                                                              print(end=" ")
                                                                                           print("I" + str(root.data) + "I<")
                                                                                           print2DTree(root.left, space)
def PrintLeaf(root):
                                                                                         def myDel(root):
  s1 = []
  s2 = []
                                                                                           while (1):
                                                                                              delTarget = int(input('Del Node: '))
root = delNode(root, delTarget)
  s1.append(root)
                                                                                              print2DTree(root)
  while len(s1) != 0:
     curr = s1.pop()
     if curr.left:
                                                                                         def myAdd():
        s1.append(curr.left)
                                                                                           while (1):
     if curr.right:
                                                                                              myNode = input('Add Node:')
        s1.append(curr.right)
                                                                                              if myNode.isdigit():
     elif not curr.left and not curr.right:
                                                                                                root.insert(int(myNode))
        s2.append(curr)
                                                                                                myAdd()
                                                                                              elif myNode is 'P':
  print('All leaf is')
                                                                                                pNode = int(input('Parent Node of:'))
  for i in range(len(s2)):
                                                                                                 print("The parent of node"+' '+str(pNode) +
     print(s2.pop().data)
                                                                                                     '+'is '+str(parent(root, pNode)))
  return
                                                                                              elif myNode is 'C':
                                                                                                pNode = int(input('Child Node of:'))
def height(root):
                                                                                                 child(root, pNode)
  ans = 0
                                                                                              elif myNode is 'L':
  queue = collections.deque()
                                                                                                PrintLeaf(root)
  if root is None:
                                                                                              elif myNode is 'S':
pNode = int(input('Sibling Node of:'))
     return ans
                                                                                                print('The sibling node '+str(pNode) +
    ' is '+str(sibling(root, pNode)))
  queue.append(root)
                                                                                              else:
                                                                                                myDel(root)
  while queue:
     currSize = len(queue)
     while currSize > 0:
                                                                                         while (1):
        currNode = queue.popleft()
                                                                                           myRoot = input('Add Root:')
        currSize -= 1
                                                                                           if myRoot.isdigit():
                                                                                              root = Node(int(myRoot))
print2DTree(root)
        if currNode.left is not None:
           queue.append(currNode.left)
                                                                                              myAdd()
        if currNode right is not None:
                                                                                           else:
           queue.append(currNode.right)
                                                                                              print('Only INT!')
                                                                                         while (1):
     ans += 1
                                                                                           myAdd()
  return ans-1
```

```
Add Root:10

|10|<

Add Node:20

|20|<

|10|<

Height of tree is: 1

Add Node:30

|30|<

|20|<

|10|<

Height of tree is: 2

Add Node:
```

```
Add Root:10
|10|<
Add Node:20
|20|<
|10|<
Height of tree is: 1
Add Node:30
|30|<
|20|<
|10|<
Height of tree is: 2
Add Node:P
Parent Node:30
The parent of node is: 20
Add Node:
```

```
Add Root:10
|10|<
Add Node:20
|20|<
|10|<
Height of tree is: 1
Add Node:30
|30|<
|20|<
|10|<
Height of tree is: 2
Add Node:15
|30|<
|20|<
|15|<
|16|<
Height of tree is: 2
Add Node:15
|30|<
|20|<
|15|<
|10|<
Height of tree is: 2
Add Node:0
|10|<
```

```
Add Root:20
|20|<
Add Node:10
20 <
     10 <
Height of tree is: 1
Add Node:30
     30 <
20 <
     10 <
Height of tree is: 1
Add Node:L
All leaf is
10
30
Add Node:
```

```
Add Node:5

|20|<
|10|<
|5|<
Height of tree is: 1
Add Node:S
Sibling Node of:5
The sibling node 5 is 20
Add Node:
```