Candidate name: Tan Xiaochen

Centre number: Dunman High School

Index number: 6C33 07

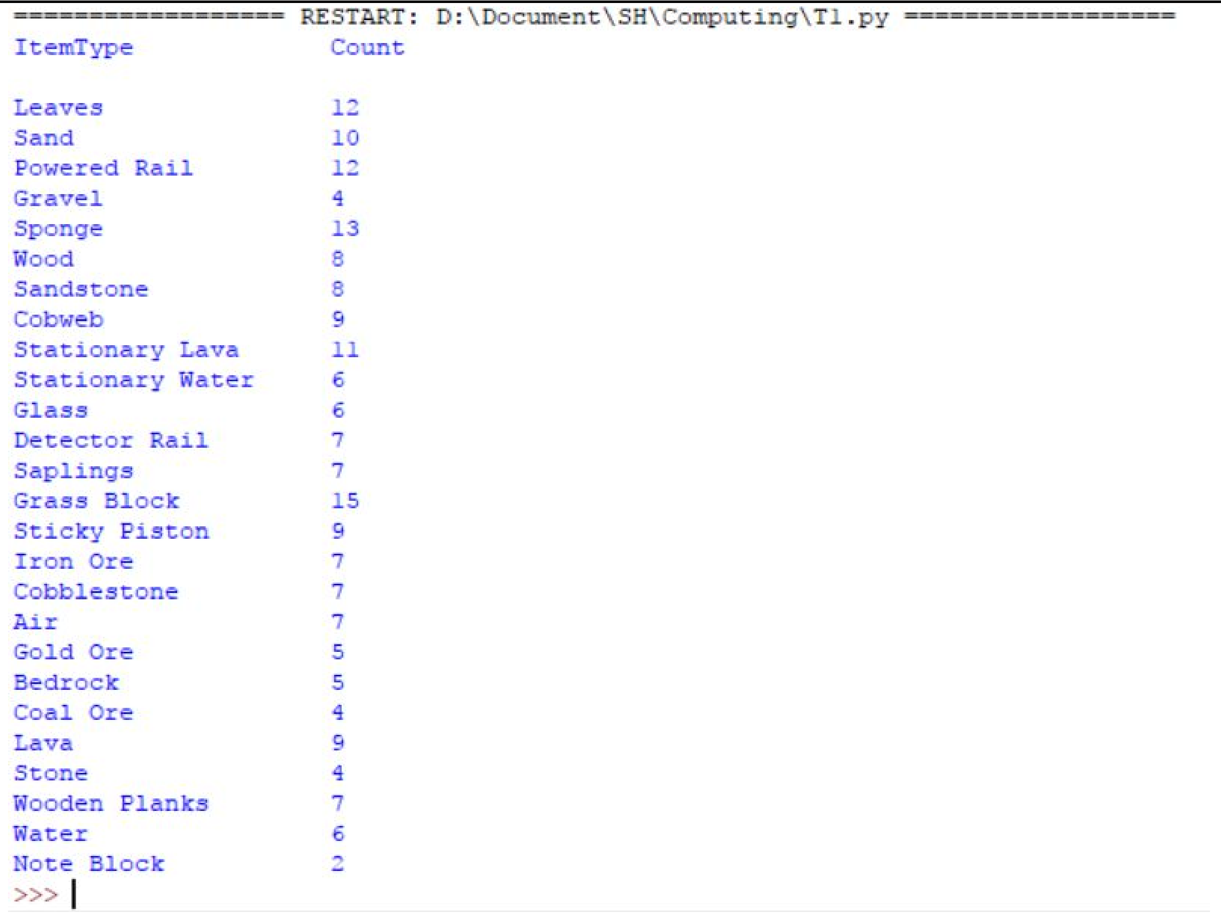
Programming language used: Python 3.2

Question 1

Evidence 1

```
#open file and read file
infile=open("INVENTORY.txt","r")
lines= infile.readlines()

#put the data read into a array
Inventory = []
for line in lines:
    Inventory.append(line.rstrip('\n')) #delete the \n at the end of the
line

#close file
infile.close()
ItemTypes = []
ItemCounts = []
for item in Inventory: #search through the item list
    if item not in ItemTypes:
        ItemTypes.append(item) # if it is new to the types list, add it into
the list

for item in ItemTypes:
    ItemCounts.append(0) #initialise the count array
for item in Inventory:
    for i in range(len(ItemTypes)):
        if item == ItemTypes[i]: #map item with the position in the type
array
            ItemCounts[i] += 1 #add count to the count array

print('{0: <20}'.format('ItemType'),'{0: <10}'.format('Count')) #using
format to print fixed length
print()
for i in range(len(ItemTypes)):
    print('{0: <20}'.format(ItemTypes[i]),'{0: <10}'.format(ItemCounts[i]))
```

Evidence 2



```
================== RESTART: D:\Document\SH\Computing\T1.py ==================
ItemType                 Count

Leaves                   12
Sand                     10
Powered Rail             12
Gravel                   4
Sponge                   13
Wood                     8
Sandstone                8
Cobweb                   9
Stationary Lava          11
Stationary Water         6
Glass                    6
Detector Rail            7
Saplings                 7
Grass Block              15
Sticky Piston            9
Iron Ore                 7
Cobblestone              7
Air                      7
Gold Ore                 5
Bedrock                  5
Coal Ore                 4
Lava                     9
Stone                    4
Wooden Planks            7
Water                    6
Note Block               2
>>> |
```

Question 2

Evidence 3

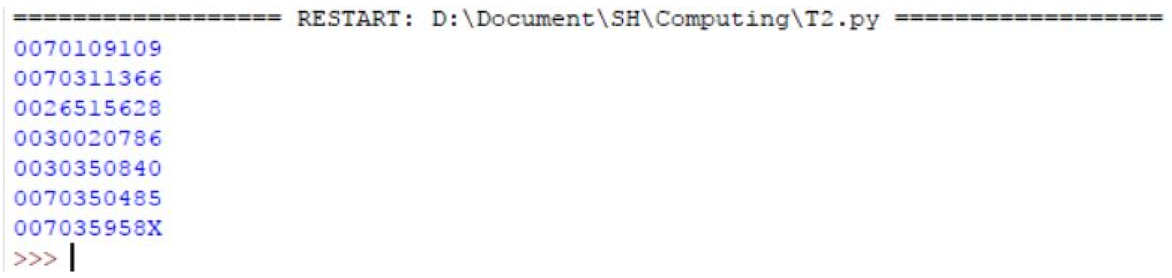
```
A: CalCheckDigit(NewNumber,Total)
B: RETURN 'x'
C: Number + CheckDigit
```

```
A: CalCheckDigit(NewNumber,Total)
B: RETURN 'x'
C: Number + CheckDigit
```

Evidence 4

```
def CalCheckDigit(Number,Total):
    if len(Number) > 1: #the current left digit is not the last digit
        Digit = int(Number[0]) #take out the left digit and transform into
integer
        Total += Digit*(len(Number)+1) #add it to total number with the
weight
        NewNumber = Number[1:len(Number)] #cut out the left digit
        CheckDigit = CalCheckDigit(NewNumber,Total) #recursive call to deal
with the next digit
    else: # the digit is the last digit
        Digit = int(Number[len(Number)-1]) #transform into integer
        Total += Digit*(len(Number)+1) #take out the left digit and
transform into integer
        CalcModulus = Total % 11
        CheckValue = 11- CalcModulus
        if CheckValue == 11:
            return str(0)
        else:
            if CheckValue == 10:
                return 'X'
            else:
                return str(CheckValue) #change back to string element
    if len(Number) == 9:
        return Number+CheckDigit #produce the final result
    else:
        return CheckDigit

#open file
infile=open("ISBNPRE.txt","r")
lines= infile.readlines()

#put the data read into a array
ISBNs = []
for line in lines:
    ISBNs.append(line.rstrip('\n')) #delete the \n at the end of the line

#close file
infile.close()

for ISBN in ISBNs:
    print(CalCheckDigit(ISBN,0))
```

Evidence 5



```
================== RESTART: D:\Document\SH\Computing\T2.py ==================
0070109109
0070311366
0026515628
0030020786
0030350840
0070350485
007035958X
>>> 
```

Question 3

Evidence 6

```
class ConnectionNode:

    def __init__(self,data =None, leftchild = 0, rightchild = 0):
#initialise with input for DataValue, LeftChild and RightChild
        self.DataValue = data
        self.LeftChild = leftchild
        self.RightChild = rightchild

class LinkedList:

    def __init__(self):
        self.RobotData = []
        self.RobotData.append(None) #occupy the zeroth position
        for i in range(1,26):
            NextNode = i+1
            if i == 25:
                NextNode = 0
            self.RobotData.append(ConnectionNode(None,NextNode))
        self.Root = 1 #list starts at position 1
        self.NextFreeChild = 1

#main
RobotRoute = LinkedList() #initialisation
```

Evidence 7

```
def FindNode(self,NodeValue):
        Found = False
        CurrentPosition = self.Root
        while Found == False and CurrentPosition <26: #check for index out
of range
            if self.RobotData[CurrentPosition].DataValue == NodeValue:
                Found = True
            else:
                CurrentPosition += 1
        if CurrentPosition >25: #if not found
            return 0
        else:
            return CurrentPosition

    def AddToRobotData(self,NewDataItem, ParentItem, ThisMove):
        if self.Root == 1 and self.NextFreeChild == 1:
            self.NextFreeChild =
self.RobotData[self.NextFreeChild].LeftChild
            self.RobotData[self.Root].LeftChild = 0
            self.RobotData[self.Root].DataValue = NewDataItem
        else:
            # does the parent exist?
            ParentPosition = self.FindNode(ParentItem)
            if ParentPosition > 0: #parent exists
                #does the child exist?
                ExistingChild = self.FindNode(NewDataItem)
                if ExistingChild > 0: #child exists
                    ChildPointer = ExistingChild
                else:
                    ChildPointer = self.NextFreeChild
                    self.NextFreeChild =
self.RobotData[self.NextFreeChild].LeftChild
                    self.RobotData[ChildPointer].LeftChild = 0
                    self.RobotData[ChildPointer].DataValue = NewDataItem
                if ThisMove == 'L': #if ThisMove indicates left
                    self.RobotData[ParentPosition].LeftChild = ChildPointer
                else:
                    self.RobotData[ParentPosition].RightChild = ChildPointer
```

Evidence 8

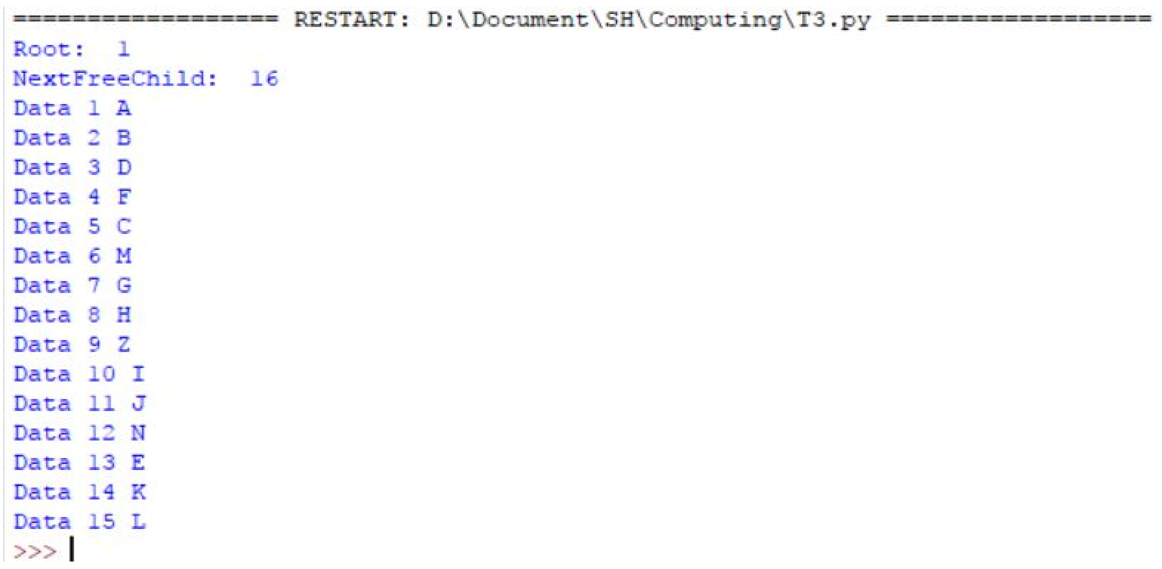
```
def OutputData(self):
        print('Root: ', self.Root)
        print('NextFreeChild: ', self.NextFreeChild)
        for i in range(1,25): #output according to index number
            if self.RobotData[i].DataValue is not None:
                print('Data',i,self.RobotData[i].DataValue)
```

Evidence 9

```
#open file and read file
infile=open("SEARCHTREE.txt","r")
lines= infile.readlines()

#put the data read into a array
NewDataItems = []
```

```
ParentItems = []
ThisMoves = []
for line in lines:
    Data = line.rstrip('\n')
    NewDataItem, ParentItem, ThisMove = Data.split(',') #split the data in
the same line
    NewDataItems.append(NewDataItem) #add data to respective list
    ParentItems.append(ParentItem)
    ThisMoves.append(ThisMove)

#close file
infile.close()

RobotRoute = LinkedList()
for i in range(20):
    RobotRoute.AddToRobotData(NewDataItems[i], ParentItems[i], ThisMoves[i])
RobotRoute.OutputData()
```

Evidence 10

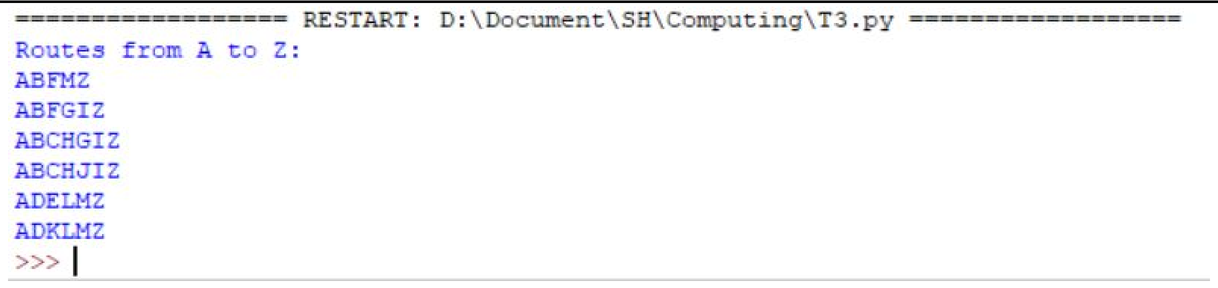


```
================== RESTART: D:\Document\SH\Computing\T3.py ==================
Root:  1
NextFreeChild:  16
Data 1 A
Data 2 B
Data 3 D
Data 4 F
Data 5 C
Data 6 M
Data 7 G
Data 8 H
Data 9 Z
Data 10 I
Data 11 J
Data 12 N
Data 13 E
Data 14 K
Data 15 L
>>> 
```

Evidence 11

```
def PreOrder(self,Pointer = None, Route = ''):
        if Pointer == None:
            Pointer = self.Root
        Route += self.RobotData[Pointer].DataValue
        if self.RobotData[Pointer].DataValue == 'Z': #terminating case if
the Z is found
            print(Route)
        if self.RobotData[Pointer].LeftChild != 0 : #if there is left child
            self.PreOrder(self.RobotData[Pointer].LeftChild,Route) #explore
route through left child
        if self.RobotData[Pointer].RightChild != 0: #if there is right child
            self.PreOrder(self.RobotData[Pointer].RightChild,Route) #explore
route through right child
```

Evidence 12



```
================== RESTART: D:\Document\SH\Computing\T3.py ==================
Routes from A to Z:
ABFMZ
ABFGIZ
ABCHGIZ
ABCHJIZ
ADELMZ
ADKLMZ
>>> |
```

Question 4

Evidence 13

```
#PROCEDURE FOR DISPLAY
DEF DISPLAY(PUZZLE):
    FOR i FROM 0 TO 3:
        FOR j FROM 0 TO 3:
            OUTPUT PUZZLE[I][J]
        END FOR
        CHANGE LINE
    END FOR

#MAIN PROBLEM
PUZZLE = [[0 FOR j FROM 0 TO 3] FOR i FROM 0 TO 3]
NUMBERS = [4,3,2,1,1,2,4,3,3,4,1,2,2,1,3,4]
FOR i FROM 0 TO 3:
    FOR j FOR 0 TO 3:
        PUZZLE[i][j] = NUMBERS[i*4+j]
    END FOR
END FOR
DISPLAY(PUZZLE)
```

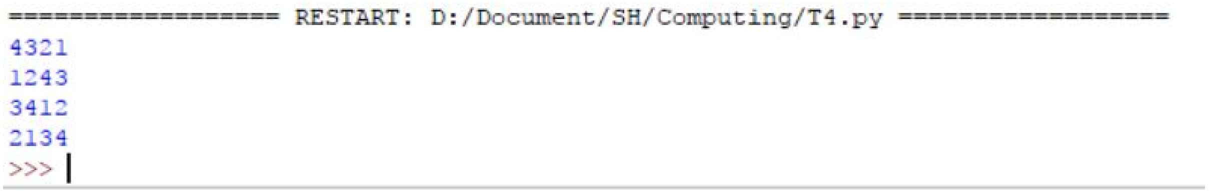
```
#PROCEDURE FOR DISPLAY
PROCEDURE DISPLAY(PUZZLE):
    FOR i FROM 0 TO 3:
        FOR j FROM 0 TO 3:
            OUTPUT PUZZLE[I][J]
        END FOR
        CHANGE LINE
    END FOR

#MAIN PROBLEM
FOR i FROM 0 TO 3:
    FOR j FROM 0 TO 3:
        PUZZLE[i][j] = 0
    END FOR
END FOR
NUMBERS = [4,3,2,1,1,2,4,3,3,4,1,2,2,1,3,4]
FOR i FROM 0 TO 3:
    FOR j FOR 0 TO 3:
        PUZZLE[i][j] = NUMBERS[i*4+j]
    END FOR
END FOR
DISPLAY(PUZZLE)
```

Evidence 14

```
def Display(Puzzle):
    for i in range(4): # from 0 to 3
        for j in range(4):
            print(Puzzle[i][j],end='')
        print()
```

```
#main
Puzzle = [[0 for j in range(4)] for i in range(4)]
Numbers = [4,3,2,1,1,2,4,3,3,4,1,2,2,1,3,4] #list for all the numbers
for i in range(4):
    for j in range(4):
        Puzzle[i][j] = Numbers[(i)*4+j] #fill in the numbers according to
index for row and column
Display(Puzzle)
```

Evidence 15



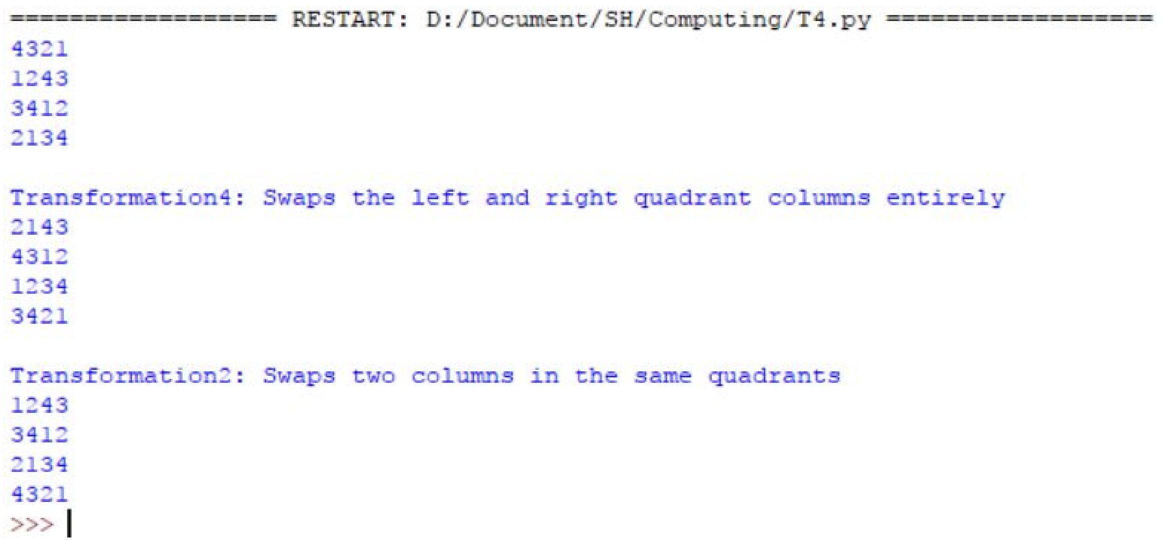
```
================== RESTART: D:/Document/SH/Computing/T4.py ==================
4321
1243
3412
2134
>>> 
```

Evidence 16

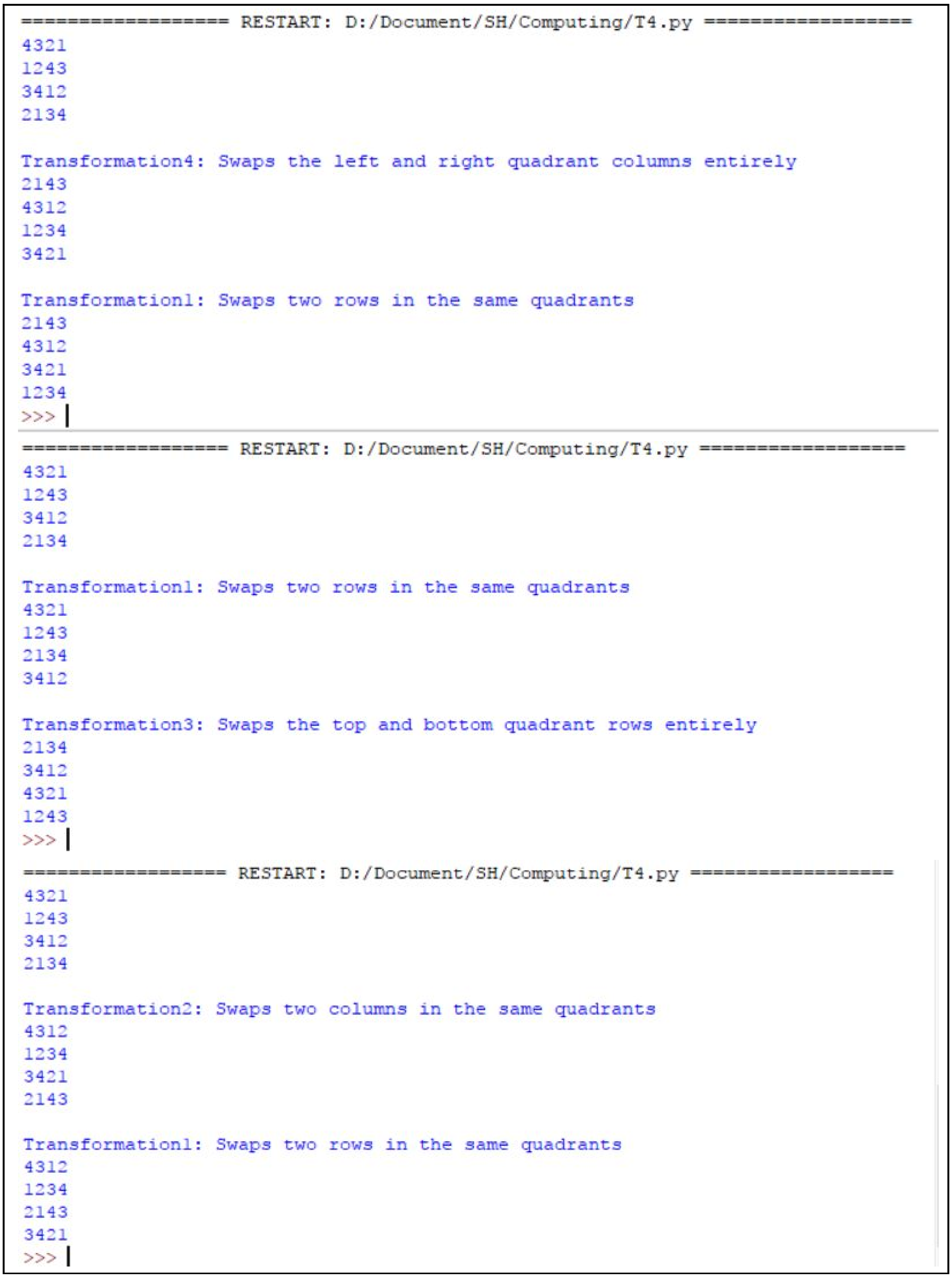
```
from random import randint

def Transformation1(Puzzle): #transformation1
    SwapQuadrant = randint(0,1) # randomly choose the rows: 0 for upper
quadrant and 1 for bottom quadrant
    for j in range(4):
        Puzzle[SwapQuadrant*2][j],Puzzle[SwapQuadrant*2+1][j] =
Puzzle[SwapQuadrant*2+1][j],Puzzle[SwapQuadrant*2][j]
    print('Transformation1: Swaps two rows in the same quadrants')
    Display(Puzzle)
    return Puzzle

def Transformation2(Puzzle): #transformation2
    SwapQuadrant = randint(0,1) # randomly choose the rows: 0 for left
quadrant and 1 for right quadrant
    for i in range(4):
        Puzzle[i][SwapQuadrant*2],Puzzle[i][SwapQuadrant*2+1] =
Puzzle[i][SwapQuadrant*2+1],Puzzle[i][SwapQuadrant*2]
    print('Transformation2: Swaps two columns in the same quadrants')
    Display(Puzzle)
    return Puzzle

def Transformation3(Puzzle): #transformation3
    for i in range(2):
        for j in range(4):
            Puzzle[i][j],Puzzle[i+2][j] = Puzzle[i+2][j],Puzzle[i][j]
    print('Transformation3: Swaps the top and bottom quadrant rows
entirely')
    Display(Puzzle)
    return Puzzle

def Transformation4(Puzzle): #transformation4
    for j in range(2):
        for i in range(4):
            Puzzle[i][j],Puzzle[i][j+2] = Puzzle[i][j+2],Puzzle[i][j]
    print('Transformation4: Swaps the left and right quadrant columns
entirely')
    Display(Puzzle)
    return Puzzle

#main
```

```
Transformation = randint(1,4) # randomly choose the first transformation
if Transformation == 1:
    Puzzle = Transformation1(Puzzle)
if Transformation == 2:
    Puzzle = Transformation2(Puzzle)
if Transformation == 3:
    Puzzle = Transformation3(Puzzle)
if Transformation == 4:
    Puzzle = Transformation4(Puzzle)
print()
NewTransformation = randint(1,4) # randomly choose the second transformation
while Transformation == NewTransformation: # make sure it is different from
the first one
    NewTransformation = randint(1,4)
Transformation = NewTransformation
if Transformation == 1:
    Puzzle = Transformation1(Puzzle)
if Transformation == 2:
    Puzzle = Transformation2(Puzzle)
if Transformation == 3:
    Puzzle = Transformation3(Puzzle)
if Transformation == 4:
    Puzzle = Transformation4(Puzzle)
```

Evidence 17



```
================== RESTART: D:/Document/SH/Computing/T4.py ==================
4321
1243
3412
2134

Transformation4: Swaps the left and right quadrant columns entirely
2143
4312
1234
3421

Transformation2: Swaps two columns in the same quadrants
1243
3412
2134
4321
>>> 
```



```
================== RESTART: D:/Document/SH/Computing/T4.py ==================
4321
1243
3412
2134

Transformation4: Swaps the left and right quadrant columns entirely
2143
4312
1234
3421

Transformation1: Swaps two rows in the same quadrants
2143
4312
3421
1234
>>> 
================== RESTART: D:/Document/SH/Computing/T4.py ==================
4321
1243
3412
2134

Transformation1: Swaps two rows in the same quadrants
4321
1243
2134
3412

Transformation3: Swaps the top and bottom quadrant rows entirely
2134
3412
4321
1243
>>> 
================== RESTART: D:/Document/SH/Computing/T4.py ==================
4321
1243
3412
2134

Transformation2: Swaps two columns in the same quadrants
4312
1234
3421
2143

Transformation1: Swaps two rows in the same quadrants
4312
1234
2143
3421
>>> 
```