210CT Coursework
Student Name: Selvachandran Thanushn
Student ID: 6785057
GitHub Repositories: https://github.com/thanushs/210CTcoursework/tree/thanushs-cw

1.

PYTHON CODE

```
import random

x = 'yes'                                                                   (1)
intList = []                                                                (1)
while x == 'yes':                      #asks the user to input the numbers  (n)
into the list one by one                                                    (n)
    y = int(input('Enter a number: '))                                      (n)
    intList.append(y)                                                       (n)
    x = input("Do you want to continue? ")                                  (1)
print(intList)                                                              (1)
count = 0                                                                   (1)
maxNumber = (len(intList)-1)                                                (1)
minNumber = 0                                                               (n)
for i in intList:                                                           (n)
    randomNumber = random.randrange(minNumber, maxNumber)                   (n*n)
    if randomNumber == count:                  #checks that the elements are not   (n*n)
swithing with themselves                                                    (n)
        randomNumber = random.randrange(minNumber, maxNumber)               (n)
    intList[count], intList[randomNumber] = intList[randomNumber], intList[count]   (1)
    count = count + 1
print(intList)
```

1+1+n+n+n+n+1+1+1+1+n+n+n^2+n^2+n+n+1 = 2(n^2) + 9n + 7

O(n^2)

The rationale behind the shuffle is a loop that uses the random number function to give the index of two different elements in a list and swithes them. It does this for every number of elements in the list.

2.

PYTHON CODE

```
x = 0                                                                                (1)
while x == 0:                                                                        (n)
    factorialInput = int(input("Enter a factorial value: "))                         (n)
    total = 1                                                                        (n)
    noOfZeros = 0                                                                    (n)
    count = factorialInput                                                           (n)
    if factorialInput > 0:                                                           (n*n)
        while count > 1:                         #loop to work out factorial of input (n*n*n))
            total = total * count * (count - 1)                                      (n*n*n))
            count = count - 2                                                        (n*n*n))
        print (total)                                                                (n*n)
        total = str(total)                                                           (n*n)
        for i in total:                          #goes through every digit and checks if (n*n*n)
they are zero and how many there are                                                 (n*n*n*n)
            if i == "0":                                                             (n*n*n*n)
               noOfZeros = noOfZeros + 1                                             (n)
    print("The number of zeros in " + str(factorialInput) + " factorial is " + str(
noOfZeros))                                                                          (1)
    print("")
```

1+n+n+n+n+n+n^2+n^3+n^3+n^3+n^2+n^2+n^3+n^4+n^4+n+1 = 2(n^4)+4(n^3)+3(n^2)+6n+2

O(n^4)

3.

PSEUDOCODE

```
value ←int
squareNumber ← 1
count ← 1
WHILE (squareNumber <= value) DO
        squareNumber ← count ^ 2
        count ← count + 1
END WHILE
IF (squareNumber > value) DO
        squareNumber ← (count – 2) ^ 2
PRINT( squareNumber)
```

PYTHON CODE

```
value = int(input("Enter a value: "))
squareNumber = 1
count = 1
while squareNumber <= value:                     #loop to calulate closest square number
    squareNumber = count ** 2
    count = count + 1


if squareNumber > value:                         #checks if the answer is bigger than out put so it goes bak one number
        squareNumber = (count - 2) ** 2
print("The highest perfect square number is " + str(squareNumber))
```

4. For this question, I have written the run time bounds for last week’s task in question 1 and 2.

5.

```
PSEUDOCODE
matrixB ← list[](2,2)                                        (1)
matrixC ← list[](2,2)                                        (1)
FUNCTION matrixMultiplication (matrixB, matrixC)             (1)
    x ← LENGTH(row of (matrixB))                             (1)
    y ← LENGTH(column of (matrixC))                          (1)
    z ←list[] (x, y)                                         (1)
    FOR i IN x DO                                            (n)
        matrixA[i] ← SUM matrixB[i] * matrixC[i]             (n)
    RETURN matrixA                                           (n)
FUNCTION matrixAddition(matrixB, matrixC)                    (1)
    IF (SIZE of matrixB = SIZE of matrixC) DO                (n)
        FOR i IN x DO                                        (n*n)
            matrixA[i] ← matrixB[i] + matrixC[i]             (n*n)
    ELSE DO                                                  (n)
        PRINT (“Both matrices are not the same size”)        (n)
    RETURN matrixA                                           (1)

FUNCTION matrixSubtraction(matrixB, matrixC)                 (1)
    IF (SIZE of matrixB = SIZE of matrixC) DO                (n)
        FOR i IN x DO                                        (n)
            matrixA[i] ← matrixB[i] + matrixC[i]             (n)
    ELSE DO                                                  (n)
        PRINT (“Both matrices are not the same size”)        (n)
    RETURN matrixA                                           (1)

matrixB ← matrixMultiplication(matrixB, matrixC)             (1)
matrixC ← (2 * matrixAddition(matrixB, matrixC))             (1)
matrixFinal ← matrixSubtraction(matrixB, matrixC)            (1)
```

1+1+1+1+1+1+n+n+n+1+n+n^2+n^2+n+n+1+n+n+n+n+n+n+1+1+1+1 = 2(n^2) + 12n + 12

O(n^2)

6.

PSEUDOCODE

```
intString ← string
finalStringList ← list
newString ← string
newStringList ← list
FOR char IN intString DO
IF char = “ ” DO
        APPEND newString TO newStringList
        newString ← “ ”
ELSE DO
        NEXT char IN newString
IF newString DO
        APPEND newString TO newStringList
count ← LENGTH(newStringList) – 1
WHILE count >= 0 DO
        x ← newStringList[count]
        APPPEND x TO finalStringList
        count = count – 1
finalString ← JOIN finalStringList
PRINT(finalStringList)
```

PYTHON CODE

```
intString = str(input("Enter the sentence: "))                      (1)
finalStringList = []                                                 (1)
newString = ""                                                       (1)
newStringList = []                                                   (1)
for c in intString:                                                  (n)
    if c == ' ':                    #ignores any spaces in the       (n*n)
string                                                               (n*n)
        newStringList.append(newString)                              (n*n)
        newString = ''                                               (n*n)
    else:                                                            (n*n)
        newString += c                                               (n)
if newString:                                                        (n)
    newStringList.append(newString)                                  (1)
count = len(newStringList) - 1                                       (n)
while count >= 0:                                                    (n)
    x = newStringList[count]        #converts the appended string    (n)
into a new list                                                      (n)
    finalStringList.append(x)                                        (1)
    count = count - 1                                                (1)
finalString = " ".join(finalStringList)  #separates the words
print(finalString)
```

1+1+1+1+n+n^2+n^2+n^2+n^2+n^2+n+n+1+n+n+n+n+1+1 = 5(n^2) +7n + 7

O(n^2)

7.

PSEUDOCODE

```
n ← integer
count ← n – 1
answer ← “ ”

FUNCTION primeCheck (n, count, answer)
    WHILE answer = “ ” AND count > 1 DO
        IF (n MOD count = 0) DO
            answer ← n + “is not a prime number”
            RETURN answer
        count ← count – 1
        IF count = 1 DO
            answer ← n + “is a prime number”
            RETURN answer
output ←primeCheck(n, count, answer)
PRINT(output)
```

PYTHON CODE

```python
n = int(input("Enter a number: "))
count = n - 1
answer = ''
def primeCheck (n, count, answer):
    while answer == '' and count > 1:       #loops until answer is outputted or ount is equal to 1
        if (n % count == 0):
            answer = (str(n) + " is not a prime number")  #checks if n is divisible by any other numbers
            return (answer)
        count = count - 1
        if count == 1:
            answer  = (str(n) + " is a prime number")
            return(answer)

output = primeCheck(n, count, answer)
print(output)
```

8.

PSEUDOCODE

```
intString ← string

FUNCTION vowels (intString)
    IF LENGTH(intString) = 0 DO
        RETURN intString
    ELSEIF (intString[0] in “aeiouAEIOU”) DO
        RETURN vowels(intString[1:])
    RETURN (intString[0] + vowels(intString[1:]))
    PRINT vowels(intString)
```

PYTHONCODE

```python
intString = str(input("Enter a string: "))
def vowels(intString):
    if len(intString) == 0:         #ends the function once the string is empty
        return intString
    elif (intString[0] in "aeiouAEIOU"):
        return vowels(intString[1:])  #returns everything exept the first element
    return (intString[0] + vowels(intString[1:]))  #brings it all together
print(vowels(intString))
```

9.

PSEUDOCODE

```
x ← "yes"
intList ← list[]
count ← 0
low ← 10
high ← 14
answer ← "False"
WHILE x = "yes" DO
    a ← integer
    APPEND a TO intList
    x ← string
SORT intList IN ascending order
WHILE LENGTH(intList) > 0 AND answer = "False" DO
    IF (LENGTH(intList) MOD 2) = 0 DO
        count ← LENGTH(intList)
    ELSE DO
        count ← LENGTH(intList) + 1
    count ← INT((count/2) – 1)
    IF intList[count] >= low and intList[count] <= high DO
```

```
                answer ← "True"
        ELIF intList[count] < low DO
                DELETE intList[:count]
        ELSE DO
                DELETE intList[count:]
PRINT answer
```

PYTHON CODE

```
x = "yes"                                                              (1)
intList = []                                                           (1)
count = 0                                                              (1)
low = 10                                                               (1)
high = 14                                                              (1)
answer = "False"                                                       (1)
while x == "yes":                                                      (n)
    a = int(input("Enter a number: "))                                 (n)
    intList.append(a)                                                  (n)
    x = input("Do you want to continue? ")                             (n)
intList.sort(key=int)                                                  (1)
print(intList)                                                         (1)
while len(intList) > 0 and answer == "False":                          (n)
    if (len(intList) % 2) == 0:             #checks the midpoint       (n*n)
value in the list                                                      (n*n)
        count = len(intList)                                           (n*n)
    else:                                                              (n*n)
        count = len(intList) + 1                                       (n)
    count = int((count / 2) - 1)                                       (n*n)
    if intList[count] >= low and intList[count] <= high:  #checks      (n*n)
where the midpoint lies                                                (n*n)
        answer = "True"                                                (n*n)
    elif intList[count] < low:                                         (n*n)
        del intList[:count]                                            (n*n)
    else:                                                              (1)
        del intList[count:]
print(answer)
```

1+1+1+1+1+1+n+n+n+n+1+1+n+n^2+n^2+n^2+n^2+n+n^2+n^2+n^2+n^2+n^2+n^2+1 = 10(n^2) + 6n + 9

O(n^2)

10.

PYTHON CODE

```python
numberList = []
count = 0
finalNumberList = []
finalNumberList2 = []
x = 'yes'
while x == 'yes':
  numberList.append(int(input("Enter an integer: ")))
  x = str(input("Do you want to add another number? "))
print(numberList)
for i in range(len(numberList)):            #repeats for length of list
    finalNumberList.append(numberList[i])
    if i ==len(numberList)- 1 or numberList[i] > numberList[i+1] : #check for where the value of i is
        if len(finalNumberList) > len(finalNumberList2):  #compares the two lists for their lengths
            finalNumberList2 = finalNumberList
            finalNumberList=[]
        else:
            finalNumberList2 = finalNumberList2
        print(finalNumberList, finalNumberList2)
print(finalNumberList2)
```

11.

PYTHON CODE

```python
class Node(object):
    def __init__(self, value):
          self.value=value
          self.next=None
          self.prev=None

class List(object):
    def __init__(self):
        self.head=None
        self.tail=None
    def insert(self,n,x):
        if n!=None:
            x.next=n.next
            n.next=x
            x.prev=n
            if x.next!=None:
                x.next.prev=x
        if self.head==None:
            self.head=self.tail=x
            x.prev=x.next=None
        elif self.tail==n:
            self.tail=x
    def display(self):
        values=[]
        n=self.head
        while n!=None:
            values.append(str(n.value))
            n=n.next
        print "List: ",",".join(values)
    def delete(self, n):
        if n.prev != 0:                            #if the node is empty then it moves on to the next node
            n.prev.next = n.next
        else:
            self.head = n.head
        if n.next != 0:                   #replaces the node with the next one if it is larger
            n.next.prev = n.prev
        else:
            self.tail = n.prev

if __name__ == '__main__':
    l=List()
    l.insert(None, Node(4))
    l.insert(l.head,Node(6))
    l.insert(l.head,Node(8))
    l.display()
```

12.

PYTHON CODE

```
class BinTreeNode(object):

    def __init__(self, value):
        self.value=value
        self.left=None
        self.right=None



def tree_insert( tree, item):
    if tree==None:
        tree=BinTreeNode(item)
    else:
        if(item < tree.value):
            if(tree.left==None):
                tree.left=BinTreeNode(item)
            else:
                tree_insert(tree.left,item)
        else:
            if(tree.right==None):
                tree.right=BinTreeNode(item)
            else:
                tree_insert(tree.right,item)
    return tree

def postorder(tree):
    if(tree.left!=None):
        postorder(tree.left)
    if(tree.right!=None):
        postorder(tree.right)
    print (tree).value


def in_order(tree):
    x = []
    while(tree != null):                   #if the tree node is empty then it moves on the left hand side
        if (tree != null):
            x.push(tree)
            tree = tree.left
        else:
            tree = x.pop()                 #otherwise it moves the right and deletes that value in the new list
            visit(tree)
            tree = tree.right

```

```python
if __name__ == '__main__':

    t=tree_insert(None,6);
    tree_insert(t,10)
    tree_insert(t,5)
    tree_insert(t,2)
    tree_insert(t,3)
    tree_insert(t,4)
    tree_insert(t,11)
    in_order(t)
```

13.

```
PSEUDOCODE

originalVertex ← []

vertices ← []

vertex ← character

FUNCTION newVertex (vertex)

    IF vertex != originalVertex DO

        vertices ← vertex

        RETURN True

    ELSE DO

        RETURN False

FUNCTION newEdge (edge, originalVertex)

    IF edge NOT IN originalVertex DO

        APPEND edge TO vertices.neighbour

    ELSE DO

        RETURN FALSE

FUNCTION printGraph (vertices, edge)

    FOR i IN vertices DO

        PRINT(vertices[i], edge)


CALL printGraph(vertices, edge)
```

For the python code, I had attempted to do the code but I kept on getting more and more errors and I couldn’t solve it in the time given. Here is what I could do and hope that it still counts for some marks. Thanks

PYTHON CODE

```python
edgeList = ['AB', 'AE', 'BF', 'CG', 'DE', 'DH', 'EH', 'FG', 'FI', 'FJ']
class vertex:
    def init(self, n):
        self.name = n
        self.neighbours = list()
    def addNeighbour(self, v):
        if v not in self.neighbours:
            self.neighbours.append(v)
            self.neighbours.sort()

class graph:
    vertices = {}

    def newVertex(self, vertex):
        if isinstance(vertex, Vertex) and vertex.name not in self.vertices:
            self.vertices[vertex.name] = vertex
            return True
        else:
            return False

    def newEdge(self, u, v):
        if u in self.vertices and v in self.verties:
            for key, value in self.vertices.items():
                if key == u:
                    value.newNeighbour(v)
                if key == v:
                    value.newNeighbour(u)
            return True
        else:
            return False

    def printGraph(self):
        for key in sorted(list(self.vertices.keys())):
            print(key + str(self.vertices[key].neighbours))


graph.printGraph()
```