210CT Coursework
Student ID - 5623952

Link to GitHub repo - https://github.com/JoeParkes/210CT

Table of contents

Question 1 - Write a Function that randomly shuffles an array of integers and explain the rationale behind its implementation

```
array = [5,2,3,6,1,12]

print "This is the noraml List: - "
print (array)
temp = '0'
lastvalue = 2

temp = array[0]
array[0] = array[lastvalue]
array[lastvalue] = temp

print "This is the changed List: - "
print(array)
```

This is the code that takes an array which has been defined ‘array’ at the top and then we have it print saying to the user that the list hasn't been changed and that this is the ‘Normal List’. Then we print that array. So when we create the temp = array[0] this means that we are creating a temporary array and then we are assigning the value 0 in the array. Then we say that the new array[0] we then bring the last value from the normal array and put this assigned to the new array.

We then print the new array and say this is the changed list so the user now’s that it has been changed.

Question 2 - Count the Number of Trailing 0's in a factorial number

```
x = int(input("What is your number?:  "))

def TrailZero(x):
    Zeros = 0
    for number in range(2, x+1):
        while number > 0:
            if number % 5 == 0:
                Zeros += 1
                number = number/5
            else:
                break
    return Zeros

print ("Your number has the following number of trailing
0's")
print TrailZero(x)
```

This function takes the user input which is a number classified as an 'int' and then using this input in the function below it. We call it TrailZero and then we have it run for x which is has been defined as the users input.

So we have then defined zeros as 0 and then we create a 'for' loop. If the number is more than 0 then it shall

NEEDS COMPLETING

Question 3 - Write the pseudocode for a function which returns the highest perfect square which is less or equal to it's parameter (A positive Integer).Implement this is a programming language of you choice.

```
N = INT INPUT "ENTER NUMBER HERE"

FUCTION PERFECTSQUARENUMBER(N):
    X = N
    WHILE This is True:
        IF X to the power of 2 is EQUAL and LESS Then N:
            OUTPUT "The Highest pefect sqaure of .. or less
is .. "

        OUTPUT x**2

    ELSE
        x -= 1
```

This pseudocode shows that we take an integer from the user and then put this into the function called "PerfectSqaureNumber" of which this then goes through a while loop. This then says that 'if x is = or less < N' then it shall print that the highest perfect square is .. (Which means that we are calling what the user typed in which could be 55 if that was what the user has typed in). and the or less bit is where we have the sum that the function has calculated. Which for 55 would be 49 which is 7 x 7.

```
n = int(input("What is your number?:  ")) # This is for the
user to input the data that they want


def perfectsquarenumber(n):
    x = n #this assigns the value of x to n.
    while True:
        if x**2 <= n: # If x to the power of 2 is equal or
less than n then it shall run the following script

            print ("The highest perfect square of %s or less
is %s" %(n,x**2)) # display's the result

            return x**2 # return the value so that it can be
used outside of the function
        else:
            x -= 1

perfectsquarenumber(n)
```

We have got the user input as n and then we call it in the function. The we have a function that says if x to the power of 2 and less than and equal to n then it shall print the statement. And shall print out the outcome of what was calculated.

Question 4 - Look back at last weeks tasks and describe the run-time bounds of these algorithms Using Big 'O' Notation.

Question 1 - Big O and Time Complexity

```
array = [5,2,3,6,1,12]                                  1

print "This is the noraml List: - "                     1
print (array)                                           1
temp = '0'                                              1
lastvalue = 2                                           1

temp = array[0]                                         n
array[0] = array[lastvalue]                             n
array[lastvalue] = temp                                 n

print "This is the changed List: - "                    1
print(array)                                            1
```

This run-time bound is O(N) because it's proportional to the size of the data input. In this case it shall grow if you put more in the array. Therefore taking longer.

Question 2 - Big O and Time Complexity

```
x = int(input("What is your number?:  "))               1

def TrailZero(x):                                       1
    Zeros = 0                                           1
    for number in range(2, x+1):                        n
        while number > 0:                               n
            if number % 5 == 0:                         n?
                Zeros += 1                              n?
                number = number/5                       n?
            else:                                       n
                break
    return Zeros                                        1

print ("Your number has the following number of trailing   1
0's")
print TrailZero(x)                                      1
```

The Run-time bound is O(n2) because we are taking a number running a mathematical equation with it meaning it's performance is proportional to the size of the input

Question 5 - Write the Pseudocode corresponding to functions for addition, subtraction and multiplication of two matrices and then compute A=B*C -2*(B+C) Where B and C are two quadratic matrices of order n. What is the Run time?

Couldn’t do this one

Question 6 - Write the Pseudocode and code for a function that reverses the word's in a sentence. Give the Big 'O' Notation

```
SENTENCE = "This is awesome!"

FUCTION REV(SENTENCE)
    if SENTENCE is equal to ""
        OUTPUT SENTENCE

    ELSE
        OUTPUT REV(SENTENCE[1:]) + SENTENCE[0]

    OUTPUT "This is the sentence muddled in reverse"
    OUTPUT REV(SENTENCE)
```

This is pseudocode for creating a function that reverses a word in a sentence. This enables us to create the sentence which in this case is This is Awesome and it should return the reverse of this.

So if the sentence is blank then it shall just output the sentence that was there.

Otherwise it shall Reverse the sentence by changing the position.

Function implemented in a Python Language

```
sentence = "This is Awesome!"

def rev(sentence):
    if sentence == "":
        return sentence
    else:
        return rev(sentence[1:]) + sentence[0]

print("This is the sentence muddled up in reverse: ")
print rev(sentence)

rev(sentence)
```

Question 7 - Write a recursive function (both Pseudocode and Code) to check if a number is prime.

Pseudocode of Question 7

```
N = USER INPUT

VARIABLE = 0
i = 0

for i in Range(2, n-1):
    if N % 1 equal to 0:
        VARIABLE = VARIABLE +1

        break

    if VARIABLE equal to 0:
        OUTPUT "This is a prime number"

    ELSE
        OUTPUT "This is not a prime number"
```

Function implements in python language

```
n = int(input("What is your number?:  “)) #User Input (INT)

variable = 0
i = 0
for i in range(2,n-1) : #Loop

    if((n%i) == 0) : #If n is assigned to i and equal to 0
        variable = variable +1 #Variable shall be assigned to
variable +1

        break
if(variable == 0) :
    print("This is a prime number");

else :
    print("This is Not a prime number" );
```

This calculates if a number is prime by using 2, n-1 which calculates if it can be divisible by itself or 1.

Question 8 - Write a recursive function that (pseudocode and code) that removes all vowels from a given string

Code in python

```
word = str(input("What is your word?: “)) # User Input

def remove_vowels(word):


    if not word: # if there isn’t a word
        return word

    elif word[0] in "aeiouAEIOU": # first character is vowel
        return remove_vowels(word[1:]) # skip first character
and process rest

    return word[0] + remove_vowels(word[1:]) # return first
character and process rest


print ("We have removed the Vowels from the word :" , word ,
("\n" "The none vowels shall appear below"))

print (remove_vowels(word[1:]))

remove_vowels(word)
```

Pseudocode for Question 8

```
WORD = STR INPUT

FUCTION Remove_Vowels(WORD)
    if no word:
        OUTPUT WORD

    else if WORD [0] in "AEIOU"
        OUTPUT Remove_Vowels(WORD[1:])

    OUTPUT WORD[0] + remove_vowels(WORD[1:)

    OUTPUT ("we have removed the vowels from your word: " ,
WORD , (NEWLINE "The None Vowles shall appear below"))

    OUTPUT remove_vowels(word[1:])
```

Week 4 : Question 1 - Adapt the binary search tree algorithm so that instead of outputting whether specific value was found, it outputs whether a a value within an interval was found. Write the pseudocode and code and give the time complexity of the algorithm using big ‘o’ notation.

Pseudocode

```
ARRAY = [1,2,..]

FUNCTION BINARYSEARCH(A, l, u)
    FIRST = 0
    LAST = len(ARRAY) -1

    LOOP WHILE LAST GREATER THAN FIRST:
        Midpoint = FIRST + (LAST-FIRST)/2

    IF A[Midpoint] is GREATER THAN l and A[Midpoint] less
THAN u:
        RETURN TRUE

    ELSE IF A[Midpoint] less THAN u:
        FIRST = Midpoint +1

    ELSE IF A[Midpoint] greater than u:
        LAST = midpoint -1

    ELSE:
        RETURN FALSE
```

Python Code

```
A = [1,2,3,4,5,6,7,8,9]                                    1

def BinarySearch(A, l, u):                                 1
    first = 0                                              1
    last = len(A)-1                                        1

    while last > first:                                    n
        midpoint = first + (last - first)/2                n
        if A[midpoint] > l and A[midpoint] < u:            n?
            return True                                    1

        elif A[midpoint] < u:                              n?
            first = midpoint +1                            n

        elif A[midpoint] > u:                              n?
            last = midpoint -1                             n

        else:                                              n
            return False                                   1


print (("This is the list we are searching from", A))
print BinarySearch(A,1,3)
```

The time complexity is O(LogN) because of the fact the the algorithm will gain in performance because you will be halting the possible entries each time.Because we are only looking a certain values we are therefore cutting the time to search for values.

Question 9 - Given a sequence of n integer numbers, extract the sub-sequence of maximum length which is in ascending order.

Couldn't do this one

Question 10 - Based on the python code supplies in class as a starting point, implement the double linked list node delete function

```
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 delete(self, N):
        if N.prev != 0:
            N.prev.next = N.next
        else:
            self.head = N.next

        if N.next != 0:
            N.next.prev = N.prev
        else:
            self.tail = N.prev


    def display(self):
        values=[]
```

```
        n=self.head
        while n!=None:
            values.append(str(n.value))
            n=n.next
        print "List: ",",".join(values)


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

Question 11 - Implement TREE-SORT algorithm in a language of your choice, but make sure that the INORDER function is implemented iteratively

Could not do this one

Question 12 - Write the Pseudocode for an unweighted graph structure. You either use an adjacency matrix or adjacency list approach. Also write a function to add a new node and function to add an edge. Following that, implement the graph you have designed in the programming language of your choice. You may use your own linked list from previous labs.

```
Graph = {
     'A': set(['B', 'C']),
     'B': set(['A', 'D', 'E']),
     'C': set(['A', 'F']),
     'D': set(['B', 'G']),
     'E': set(['B', 'F', 'G']),
     'F': set(['C', 'E']),
     'G': set(['E', 'D'])}


class node:
     def __init__(self, node, edges = []):
          self.label = node
          self.edges = edges


     def __str__(self):
          return 'Node(' + str(self.label) + ", " +
str(self.edges) +")"


class Graph(node):
     def __init__(self):
          self.nodes = {}


     def addnode(self,name):
          self.node[name] = node(name)

     def addedge(self, source, target):
          self.nodes[source].edges = self.node[source].edges +
[target]

     def dot(self):
          for nodename in self.node:
               for edge in self.node[nodename].edges:
                    print(str(nodename) + " -> "+ str(edge))
```

```
if __name__ == '__main__':
    G = Graph()
    G.addnode(A)
    G.addedge(A)
    G.addnode(B,C)
    G.addnode(D)
    G.addedge(B,C)
    G.addnode(E)
    G.addedge(E,F)
    G.addnode(G)
    G.addedge(G,D)
```

Pseudocode for Question 12 -

```
CLASS Graph(NODE):

    FUNCTION__init__(self):
        self.node = {}

    FUNCTION addNode(self, name):
        self.node[name] = node[name]

    FUNCTION completedEdge(self, source, target):
        self.node[source] finds the edge and is = to
self.node[source] edge + [target of node]
```

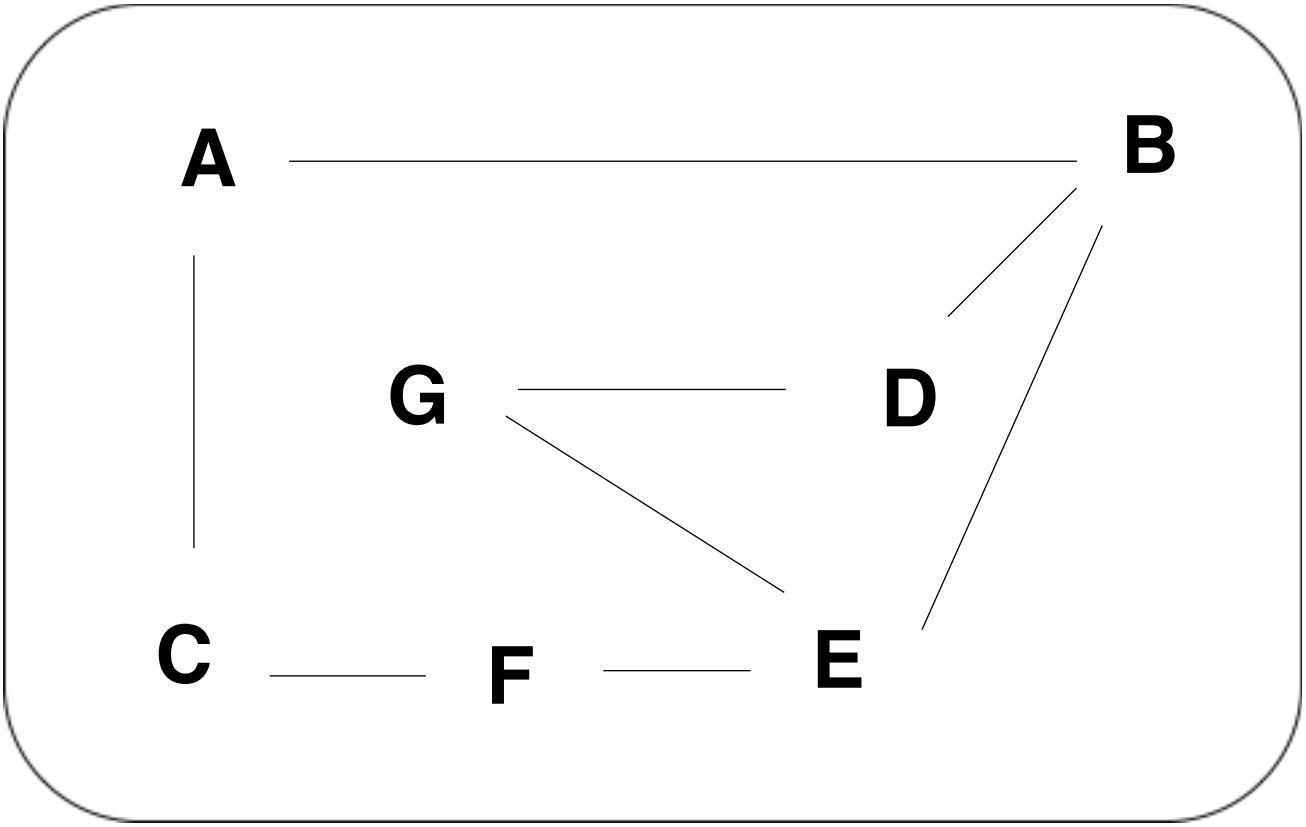
Question 13 - Implement BFS and DFS Traversals for the above graph. Save the nodes traversed in a sequence to a text file.

```
graph = {
    'A': set(['B', 'C']),
    'B': set(['A', 'D', 'E']),
    'C': set(['A', 'F']),
    'D': set(['B', 'G']),
    'E': set(['B', 'F', 'G']),
    'F': set(['C', 'E']),
    'G': set(['E', 'D'])}

def dfs(graph, start):
    visited, stack = set(), [start]
    while stack:
        vertex = stack.pop(0)
        if vertex not in visited:
            visited.add(vertex)
            stack.extend(graph[vertex] - visited)
        return visited

print dfs(graph, 'A') # {'F', 'C', 'E', 'A', 'D', 'B', 'G'}

dfs(graph, 'A')
```



The code above shows that the graph has 7 nodes of which these are A, B, C, D, E, F and G. The DFS is only called once so it keeps track of all the nodes within the graph. I have created it to be able to conduct a depth first search, this has been created by the function that has been create named dfs (depth first search). The code works and has given an output of –

Set([‘A’, ‘C’, ‘B’, ‘E’, ‘D’, ‘G’, ‘F’])

This is the output which is shown above and it has put into the order of which it got each item. When this is put into a tree it should look like the diagram below.

Question 14 - Implement Dijkstra's algorithm for a weighted graph structure

Couldn't do this one