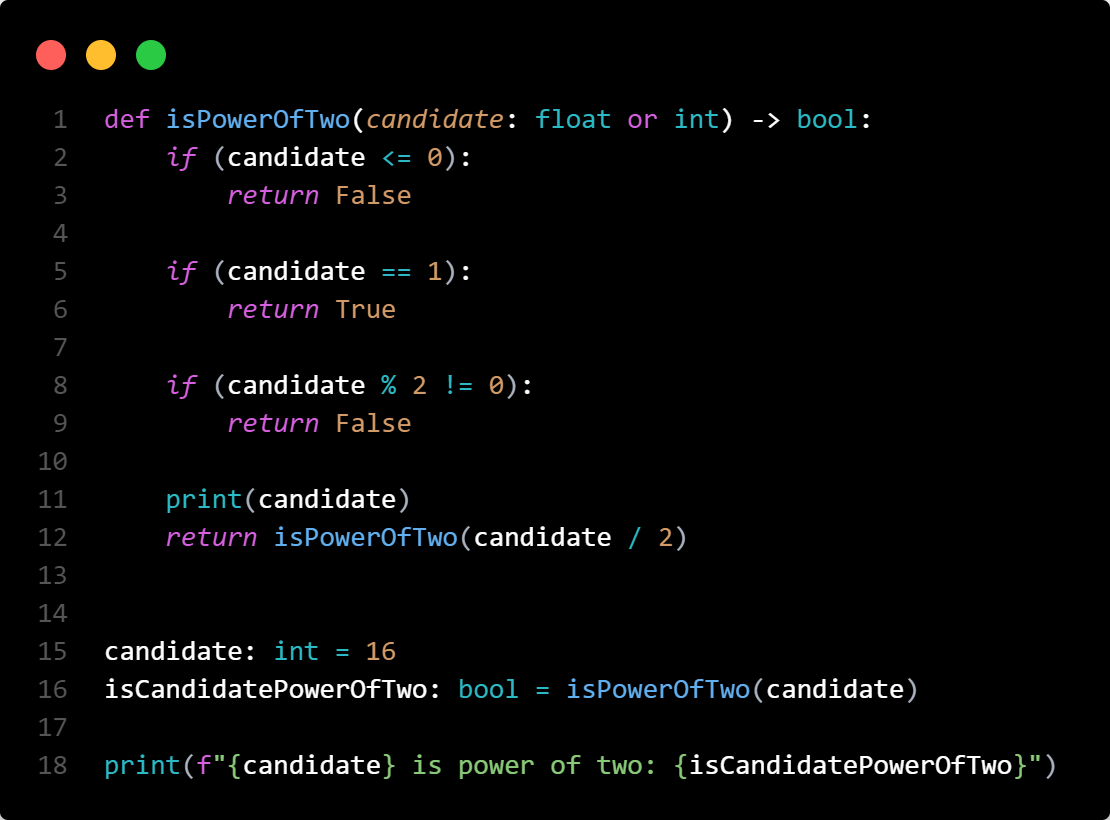
CO1107 Mock Test

Recursion

**Question 1.**

The isPowerOfTwo is a recursive function that is used to check if a candidate number is a power of two

What is the output of this code?



c)

16

8

4

2

16 is a power of two: True

d)

2

4

8

16

16 is a power of two: True

a)

16

8.0

4.0

2.0

16 is a power of two: True

b)

2.0

4.0

8.0

16

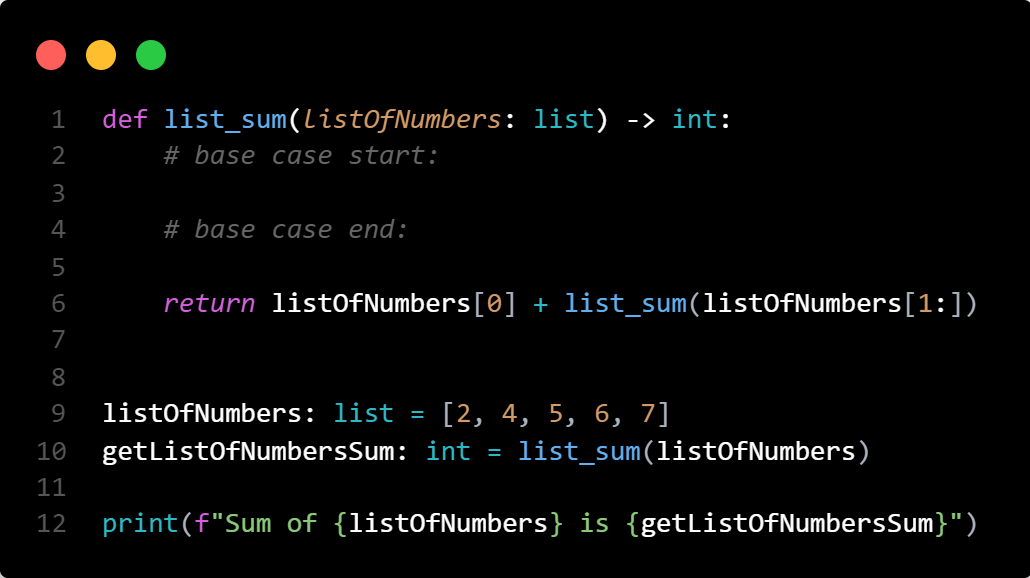
16 is a power of two: True

**Question 2**

List\_sum is a recursive function that return the sum of a list

Eg the sum of the list [2,4,5,6,7] would be 24

What would be the base case for this function?

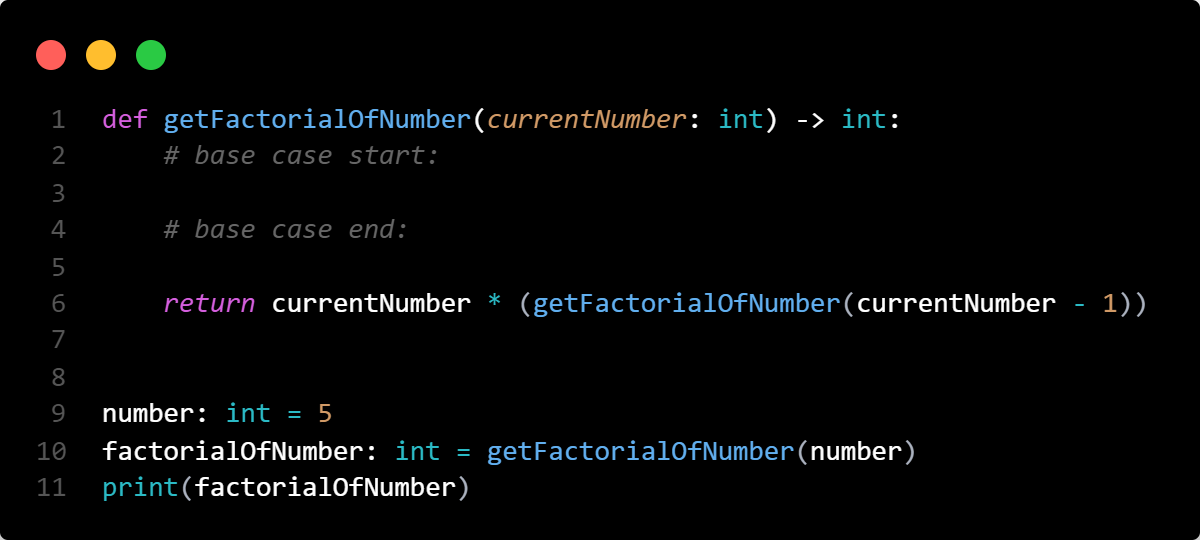


**Question Three**

This Python function get the factorial of a non-negative integer.

Eg: getFactorialOfNumber(5) would return 120

what would be the base case for the factorial function?



**Question Four**

The power function returns the base integer to power of its exponent

Eg: power(3, 4) would return 81

What would be the output of this code?

a)

3 ^ 4 = 81

3 ^ 3 = 27

3 ^ 2 = 9

3 ^ 1 = 3

d)

3 ^ 0 = 1

3 ^ 1 = 3

3 ^ 2 = 9

3 ^ 3 = 27

3 ^ 4 = 81

c)

3 ^ 4 = 81

3 ^ 3 = 27

3 ^ 2 = 9

3 ^ 1 = 3

3 ^ 0 = 1

b)

3 ^ 1 = 3

3 ^ 2 = 9

3 ^ 3 = 27

3 ^ 4 = 81

Heap:

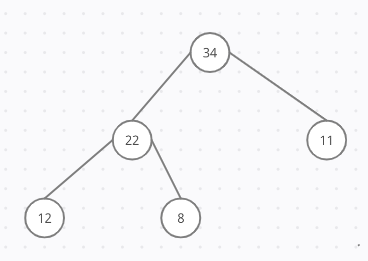
**Question 1:**

What are the properties of the following?

* A balanced binary true
* Min-heap
* Max-heap

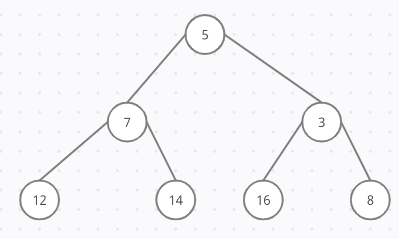
**Question 2:**

Insert 7 into this heap and draw the final tee:



**Question 3:**

Insert 6 into this heap and draw the final tree



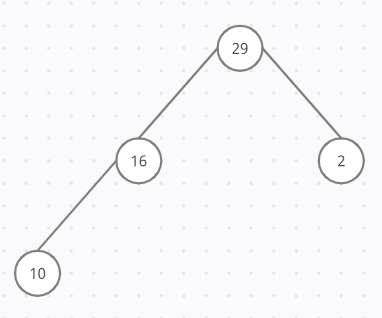
**Question 4**

Draw and create a max heap from this list:

List = [10,16,2,29]

**Question 5**

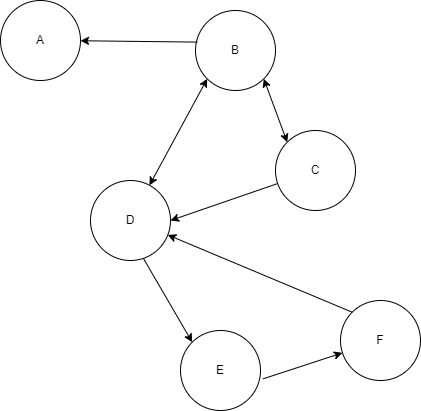
Given the following heap, state the order that the nodes are extracted in:



GRAPHS

**Question 1:**

Given the following graph, provide the adjacency matrix and the type of the graph.

****

**Question 2:**

Using the following adjacency list, create the directed and weighted graph.

Adjacency List = { “A” : [[A,1],[B,3],[D,5]],

“B” : [[A,3],[C,2], [F,7]],

“C” : [[D,1], [E,3], [F,2]],

“D” : [],

“E” : [[C,3], [F,1]],

“F” : [[E,1]] }

**Question 3:**

Complete the following Breadth-first searchalgorithm so that it prints all the nodes from the chosen start node.

def bfs(graph, node):

    visited= set()

    queue = [node]

    while not (queue==[]):

        m=queue[\_\_\_]

        queue=queue[1:]

        \_\_\_\_\_\_\_\_\_\_\_

        visited.\_\_\_\_\_(m)

        for neighbour in graph[m]:

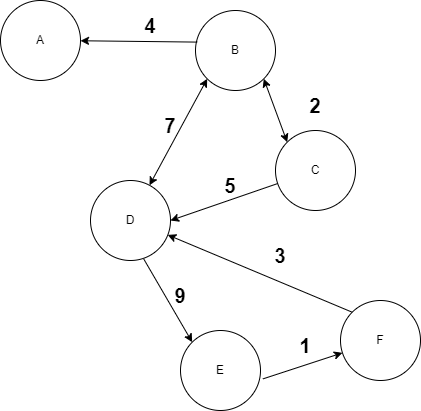
            if \_\_\_\_\_\_\_\_\_\_ not in visited:

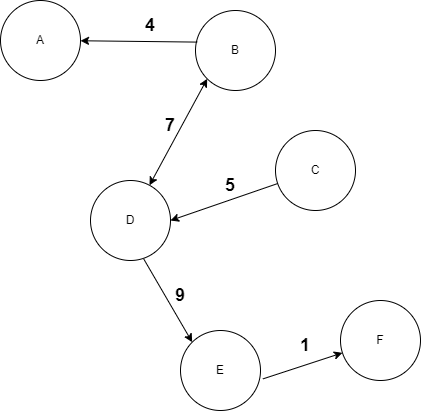
                visited.\_\_\_\_\_\_(neighbour)

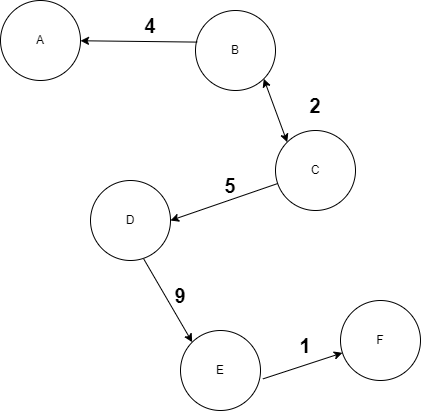
                queue.append(neighbour)

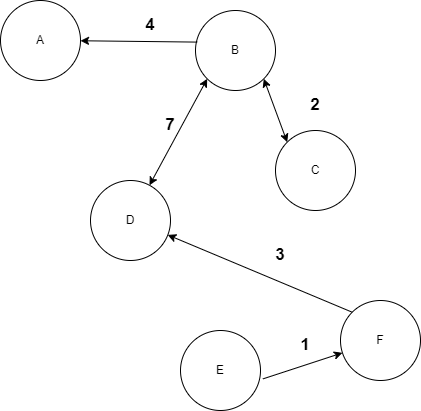
**Question 4:**

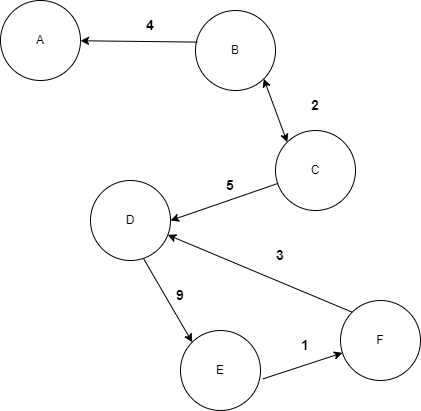
Given the following graph, choose the matching spanning tree starting from C.

****

1. 

1. 

****

2. ****

Linked List

**Question 1:**

Complete the following code to insert a new node in the linked list

def insert(self, data):

newNode = Node(data)

if(self.head):

current = self.head

while(current.next):

# code goes here

# code goes here

else:

self.head = newNode

**Question 2:**

In Linked List implementation, a node carries information regarding

1. Data
2. Data and Link
3. Link
4. None of the above

**Question 3:**

Spot the error in the code

def traverse(self):

curr=self.head

while curr!=None:

print(curr.data)

curr.next

**Question 4:**

Write an \_\_init\_\_ function to create a node in linked list

class Node:

# code goes here  
  
  
myNode = Node(10)  
print("The data in the node is:", myNode.data)

**Question 5:**

Complete the method to count the number of elements in a linked list

def getCount(self):

temp = self.head

count = 0

#code goes here

Sequential Search:

**Question 1:**

If each for-loop iteration takes one millisecond, how long will the following code take before terminating or exiting?

Names  = [ "John", "Ali", "Tom" ,"Reza", "Chris", "James" ]

Search = [ "Ali", "Reza", "Tom", "James" ]

target  = Search[3]

for name in Names:

if name == target:

        break

1. 5
2. 8
3. 6
4. 4
5. 7

**Question 2:**

What is the output of the following code?

Colours = ["Red", "Blue" , "Yellow", "Grey", "Orange"]

count = 1

for i in Colours:

if i == "Orange":

count+=1

print ("We have found a match")

else:

count -= 1

print ("There is no match!")

print(count)

Sorting Algorithms:

**Question 1:**

What is the order of the list after the fourth iteration?

Numbers = [45, 234, 78, 91, -90 ,69 , 33 , 0 , -2 , 23]

def sortList(aList):

    listLength = len(aList)

    for item in range(listLength-1):

        minNumber = item

        for num in range (item + 1, listLength):

            if aList[minNumber]  >  aList[num]:

                minNumber = num

        aList[item], aList[minNumber]  = aList[minNumber] , aList[item]

        print(aList)

sortList(Numbers)

* 1. [234, 0, 91, 23, 33, 78, -90, 69, -2, 45]
  2. [-90, 33, 234, 0, 23, 91, 78, -2, 45, 69]
  3. [23, 234, 45, 91, 78, 33, 0, -2, 69, -90]
  4. [-90, -2, 0, 23, 45, 69, 33, 78, 234, 91]

**Question 2:**

What is the output of the following code?

characters = ['a' , 'v' , 'q' , 'p' , 'd' , 'x', 'l' , 'Z', 'A']

def sortIt(aList):

    listLength = len(aList)

    for item in range(listLength - 1):

        for I in range(listLength - 1):

            if aList[I] > aList[I+1]:

                aList[I] , aList[I+1] = aList[I+1] , aList[I]

sortIt(characters)

print(characters)

1. ['A', 'Z', 'a', 'd', 'l', 'p', 'q', 'v', 'x']
2. ['a' , 'v' , 'q' , 'p' , 'd' , 'x', 'l' , 'Z', 'A']
3. ['a', 'd', 'l', 'p', 'q', 'v', 'x', 'A' , 'Z']
4. ['a' , 'd' , 'q' , 'p' , 'v' , 'x', 'l' , 'A', 'Z']
5. [9]

**Question 3:**

Complete the function below to use Insertion sort.

def InsersionSort(ListB):

    listLength = len(ListB)

    for item in range(1, listLength):

        value = item

        # Write your code here

**Question 4:**

What is the order of the list after the sixth iteration?

characters = ['a' , 'v' , 'q' , 'p' , 'd' , 'x', 'l' , 'Z', 'A']

def sortIt(aList):

    listLength = len(aList)

    for item in range(listLength - 1):

        for I in range(listLength - 1):

if aList[I] > aList[I+1]:

                aList[I] , aList[I+1] = aList[I+1] , aList[I]

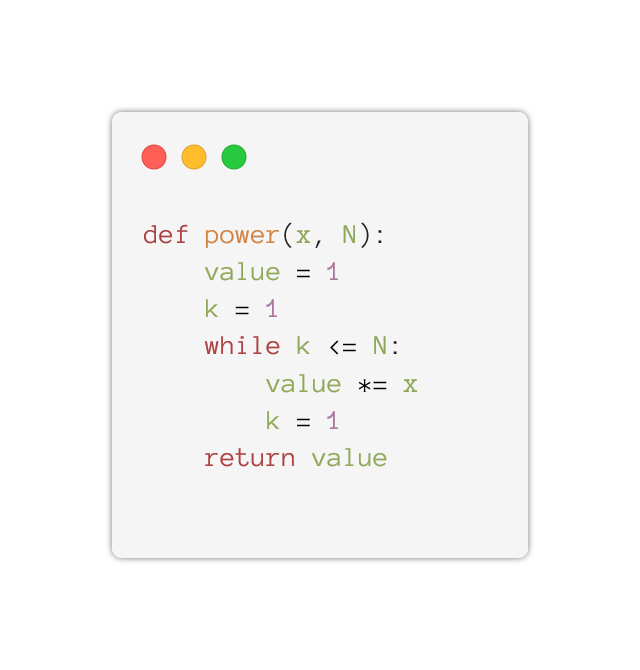
sortIt (characters)

1. ['A', 'Z', 'a', 'd', 'l', 'p', 'q', 'v', 'x']
2. ['a', 'v', 'q', 'p', 'd', 'x', 'l', 'Z', 'A']
3. ['a', 'Z', 'A', 'd', 'l', 'p', 'q', 'v', 'x']
4. ['a', 'p', 'd', 'q', 'l', 'Z', 'A', 'v', 'x']
5. Something else

Time Complexity Questions

**Question 1:**

What is the time-complexity of this code?



1. 1
2. O(N)
3. O(N2)
4. 0

**Question 2.**

What is the total running time of this code?

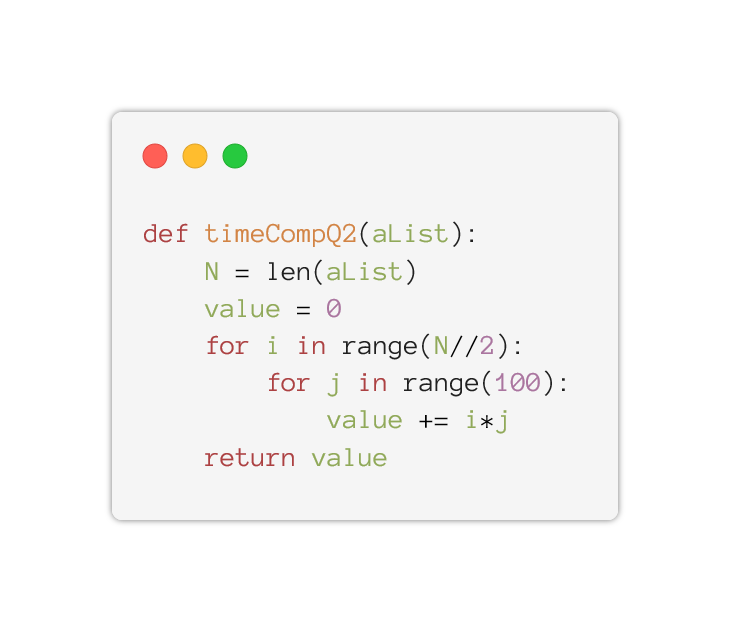
A picture containing application

Description automatically generated

1. 3 + (N + 1) + 2N
2. N+k
3. 4+(N2+2)+N
4. (N + 1) + 2N

**Question 3:**

The time complexity of below code is O(N2) because there is a loop within a loop. Is this statement True or False? Reason why.



1. True
2. False

Stack/Queue questions:

Question 1:

*Popping is an act of:*

1. Adding 2 values and returning the result
2. Transferring values between 2 stacks
3. Removing values from a stack
4. None of the above

Question 2:

*What is the minimum number of queues required for the implementation of a priority queue?*

1. 3
2. 2
3. 4
4. 10

Question 3:

*Which of the following describes a queue data structure, and which describes a stack?*

1. First in last out
2. Last in first out
3. Last in last out
4. First in first out

Question 4:

*What will be the result of this sequence of code?*

Text

Description automatically generated

1. ['Tom', 'Luke', 'James', 'John']
2. ['Tom', 'Luke', 'John', 'James']
3. ['Tom', 'James', 'John', 'Luke']
4. ['James', 'John', 'Luke', 'Tom']

Question 5:

Create a function that reverses the contents of a stack:

Trees:

**Question 1:**

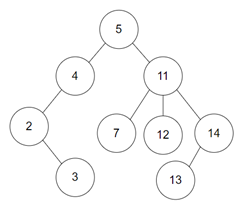
What are the properties of the following?

1. Binary tree
2. Perfect binary tree

**Question 2:**

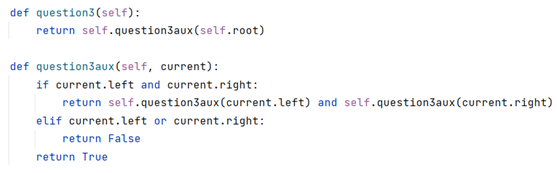
Annotate the following tree with these labels:

Node, root, leaf, parent of node 7, children of node 3, subtree with root node 11



**Question 3:**

When calling the function 'question3’ on a binary tree, when will it return True?



* 1. When the tree is perfect
  2. When all nodes have 0 or 2 children
  3. When the tree is balanced
  4. When both subtrees of the root node have an equal number of nodes

**Question 4:**

Draw a perfect binary search tree from the following list: [56,12,4,5,33,31,29]

Use 29 as the root node.

**Question 5:**

Using the following binary search tree, note down the steps taken, and the nodes visited when searching for target node:

* 1. 16
  2. 59

