

## **LEVEL 1: (4 points each)**

### **1. Differentiate between SIMD and MIMD? Explain**

MIMD stands for Multiple Instruction Multiple Data. In the SIMD design, an instruction is applied to a particular set of information or data at a constant time. SIMD stands for Single Instruction Multiple Data. While MIMD stands for Multiple Instruction Multiple Data. The SIMD architecture simultaneously performs an identical action on multiple pieces of data, including information retrieval, computation, or storage. An example is recovering multiple files at once. Processors with local memory that hold different data execute the same instruction in a synchronous manner, with inter-processor communication to assign offsets. The MIMD architecture performs several concurrent actions on multiple data sets. One example is performing different mathematical operations - such as addition and multiplication - simultaneously to solve a complex math problem with many discrete components. MIMD computing may or may not be synchronized and is becoming more common than SIMD computing. SIMD is often used for computationally intensive problems with processors performing the same operation in parallel. MIMD is often used for problems that break down algorithms into separate, independent parts, with each part assigned to a different processor for a concurrent solution.

### **2. What are the performance metrics of parallel systems?**

There are four performance metrics of parallel systems the first one is execution time. Parallel runtime is the time that elapses from the moment a parallel computation starts to the moment that processing element finished execution. And the second one is total parallel overhead. Total time collectively spent by the processing elements – running time required by the fastest known sequential algorithm for solving the same problem on a single processing element. The third one is speed up is the ration of the serial runtime of the best sequential algorithm for solving a problem to the time taken by the parallel algorithm to solve the same problem on p processing elements. And the last one is efficiency; Efficiency is a ration of speedup to the number processing element.

## **LEVEL 2: (5 points each)**

### **1. How does the parallel programming works? What do you think will be the advantage of utilizing parallel approach?**

Programming can be differentiated with larger or smaller problems. As with parallel programming, this process breakdowns larger problems into smaller problems. Multiple processors are working simultaneously that communicates with a shared memory. As per the results, they are being combined after all the process as the overall. The parallel approach takes a high rate of advantage. It conserves time and effort because of many resources that are working simultaneously. It is not just effective but it is also efficient for it process a large problem faster than we could ever imagine

2. In a right triangle, the square of the length of one side is equal to the sum of the squares of the lengths of the other two sides. Write a program that prompts the user to enter the length of the three sides of a triangle and then outputs a message indicating whether the triangle is a right triangle. The advantage of parallel computing is that due to the time savings provided by parallel computing, applications can now be run in less wall-clock time. Due to the size and/or complexity of many problems, solving them on a single that computers have available in here with this, the parallel computing is the best approach.

```
Side1 = eval(input('Enter first side here: '))  
  
Side2 = eval(input('Enter second side here: '))  
  
Side3 = eval(input('Enter third side here: '))  
  
if Side3**2 == Side2**2 + Side1**2:  
    print('This is a right triangle')  
  
else:  
  
    print('This isn\'t a right triangle')
```

Output:

```
C:\Users\sdavis4147\PycharmProjects\Dem  
Enter first side here: 2  
Enter second side here: 3  
Enter third side here: 5  
This isn't a right triangle  
  
Process finished with exit code 0
```

3. Write a program that prompts the user to input a number between 0 and 35. If the number is less than or equal to 9, the program should output the number; otherwise, it should output A for 10, B for 11, C for 12... and Z for 35.

```
a=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
b=[10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35]

def getIndex(n):
    for i in range(len(b)):
        if n==b[i]:
            return i

def getLetter(index):
    for i in range(len(a)):
        if index==i:
            return a[i]

userinput=int(input("Enter an integer between 0 and 35: "))
if userinput>0 and userinput<=35:
    if userinput<=9:
        print(userinput)
    else:
        index=getIndex(userinput)
        print(getLetter(index).upper())
else:
    print("you must input Only number 0-35")
```

Output:

```
C:\Users\sDavis4147\PycharmProjects\Demo
Enter an integer between 0 and 35: 34
Y

Process finished with exit code 0
|
```

4. Write a program that will display all numbers divisible by 3, 4 and 5 from 1-50

```
for num in range(1, 51):  
    if (num % 3) == 0 or (num % 4) == 0 or (num % 5) == 0:  
        print(num)
```

Output:

```
C:\Users\sdavis4147\PycharmProjects\DemoPandas\venv\Scripts\py  
3  
4  
5  
6  
8  
9  
10  
12  
15  
16  
18  
20  
21  
24  
25  
27  
28  
30  
32  
33  
35  
36  
39  
40  
42  
44  
45  
48  
50  
  
Process finished with exit code 0  
|
```

### LEVEL 3: (6 points each, +2 points if you get 2 LEVEL 3 questions correct.)

Create a function in Python that accepts two parameters. The first will be a list of numbers. The second parameter will be a string that can be one of the following values: asc, desc, and none. If the second parameter is "asc," then the function should return a list with the numbers in ascending order. If it's "desc," then the list should be in descending order, and if it's "none," it should return the original list unaltered.

```
y = input("type asc, desc or none: ")
res = [10, 34, 54, 65, 23, 54, 23, 3, 5, 76]
if y == "asc":
    res.sort()
    print(str(res))
if y == "desc":
    res.sort(reverse=True)
    print(str(res))
if y == "none":
    print(str(res))
```

```
C:\Users\sudavis4147\PycharmProjects\demo\demo.py
type asc desc or none: asc
[3, 5, 10, 23, 23, 34, 54, 54, 65, 76]

Process finished with exit code 0
```

## LEVEL 4: (10 points each)

1. Write a program that will generate 100 3-digit random numbers and store it in a list. The program should display the following: a. All elements in the list b. All numbers grouped by odd and even numbers c. All numbers divisible by 9. d. All prime numbers e. All numbers that contains the digit 9 (e.g 29, 91, 393, 961).

```
1 import random
2
3 prime = []
4 odd = []
5 divisible = []
6 numbers = []
7 containsNine = []
8 even = []
9
10 for x in range(100):
11     num = random.randint(100, 999)
12
13     if num % 2 == 0:
14         even.append(num)
15     else:
16         odd.append(num)
17
18     if num % 9 == 0:
19         divisible.append(num)
20
21     dividends = 0
22     for y in range(1, num):
23         if num % y == 0:
24             dividends += 1
25
26     if dividends == 2:
27         prime.append(num)
28
29     if "9" in str(num):
30         containsNine.append(num)
31
32     numbers.append(num)
33
34 print("Printing all numbers...")
35 for x in numbers:
36     print(x)
37 print("\nPrinting all odd numbers...")
38 for x in odd:
39     print(x)
40 print("\nPrinting all even numbers...")
41 for x in even:
42     print(x)
43 print("\nPrinting all numbers divisible by 9...")
44 for x in divisible:
45     print(x)
46 print("\nPrinting all prime numbers...")
47 for x in prime:
48     print(x)
49 print("\nPrinting all numbers containing the number 9...")
50 for x in containsNine:
51     print(x)
```

OUTPUT:

```
Printing all prime numbers...
Printing all numbers containing the number 9...
879
901
409
869
493
986
940
950
294
292
197
694
598
329
998
904
896
598
903
897
339
293
```

2. Given a linked list of size K, your task is to complete the function `sum_of_lastN_nodes()`, which should return the sum of last N nodes of the linked list. The function takes two arguments as input, the reference pointer of the head of the linked list and the integer N. Example: 5->10->6->4->1->12 N = 3 `sum_of_lastN_nodes(6, N)` Output: Sum of last three nodes in the linked list is 4 + 1 + 12 = 15.

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

class LinkedList:
    def __init__(self):
        self.head = None

    def sum_of_lastN_nodes(self, point, lNodes):
        current = self.head
        sum = 0

        while current is not None:
            if current.value == point:
                count = 0
                temp = current.next

                while temp is not None:
                    sum += temp.value
                    count += 1

                    if count == lNodes:
                        break

                temp = temp.next

            current = current.next

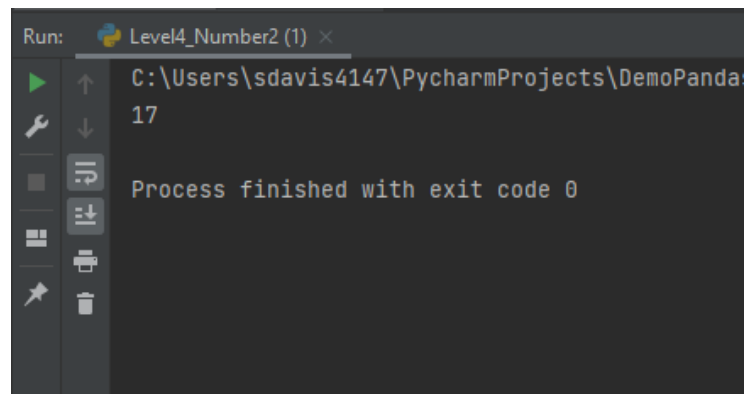
        return sum

node1 = Node(5)
node2 = Node(10)
node3 = Node(6)
node4 = Node(4)
node5 = Node(1)
node6 = Node(12)

node1.next = node2
node2.next = node3
node3.next = node4
node4.next = node5
node5.next = node6

linkedList = LinkedList()
linkedList.head = node1
print(linkedList.sum_of_lastN_nodes(6, 3))
```

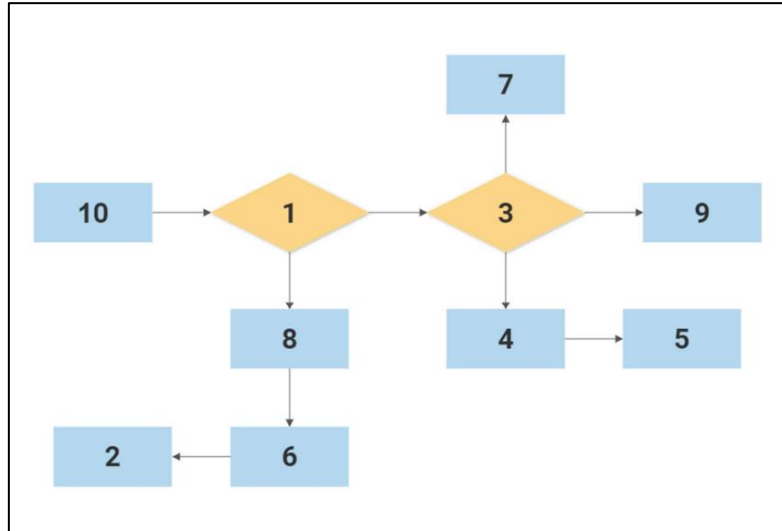
Output:



```
Run: Level4_Number2 (1) x
C:\Users\sDavis4147\PycharmProjects\DemoPanda
17
Process finished with exit code 0
```

## LEVEL 5: (20 points)

1. Flatten the given Linked list Sorting must be performed during the flattening of the linked list



```
class Node():
    def __init__(self, data):
        self.data = data
        self.right = None
        self.down = None

class LinkedList():
    def __init__(self):
        # head of list
        self.head = None

    def push(self, head_ref, data):
        new_node = Node(data)
        new_node.down = head_ref
        head_ref = new_node

        return head_ref

    def printList(self):
        temp = self.head
        while (temp != None):
            print(temp.data, end=" ")
            temp = temp.down

        print()

# An utility function to merge two sorted linked lists
def merge(self, a, b):
    if (a == None):
        return b

    if (b == None):
        return a
```

```
        result = None

        if (a.data < b.data):
            result = a
            result.down = self.merge(a.down, b)
        else:
            result = b
            result.down = self.merge(a, b.down)

        result.right = None
        return result

    def flatten(self, root):
        if (root == None or root.right == None):
            return root

        root.right = self.flatten(root.right)

        # now merge
        root = self.merge(root, root.right)

        return root

L = LinkedList()
```

```
'''
Let us create the following linked list
      7
      ^
      |
10 -> 1 -> 3 -> 9
      |   |
      v   v
      8   4 -> 5
      |
      v
'''
```



```

    |
    V
2 <- 6

'''
L.head = L.push(L.head, 9)
L.head = L.push(L.head, 3)
L.head = L.push(L.head, 2)
L.head = L.push(L.head, 1)

L.head.right = L.push(L.head.right, 7)
L.head.right = L.push(L.head.right, 4)
L.head.right.right = L.push(L.head.right.right, 8)
L.head.right.right = L.push(L.head.right.right, 5)

L.head.right.right.right = L.push(L.head.right.right.right, 10)
L.head.right.right.right = L.push(L.head.right.right.right, 6)

# flatten the list
L.head = L.flatten(L.head)

L.printList()

```

**Output:**

```

C:\Users\sDavis4147\PycharmProjects\DemoPandas\venv\Scripts\
1 2 3 4 5 6 7 8 9 10

Process finished with exit code 0

```