SK_CODING_CHALLENGE_2 – 08/06/2020

Develop the programs in any language of your choice to solve the following problems.

Activity 1:

Given an array, X of N integers, calculate and print the standard deviation. Your answer should be in decimal form, rounded to a scale of 1 decimal place. An error margin of +0.1 or -0.1 will be tolerated for the standard deviation.

Input format:
1. The first line contains an integer,N , denoting the number of elements in the array.
2. The second line contains N space-separated integers describing the respective elements of the array.

Output format:
Print the standard deviation on a new line, rounded to a scale of 1 decimal place.

Constraints:
N = [5 to 100]
x = [0 to 10000], is the index of array X

Solution:

```
n = int(input().strip())
X = [int(x) for x in input().strip().split()]

mean = sum(X) / n
variance = sum([((x - mean) ** 2) for x in X]) / n
stddev = variance ** 0.5

print("{0:0.1f}".format(stddev))
```

****

Activity 2:

You are given an array of integers, a, a[]= {1,2,3,4,,5,6….k} and a positive integer, k. Find and print the number of pairs(i, j) where i<j and a[i]+ a[j] is divisible by k.

Function Description

Complete the divisibleSumPairs function in the editor below. It should return the integer count of pairs meeting the criteria.

divisibleSumPairs has the following parameter(s):

n: the integer length of array a
a: an array of integers
k: the integer to divide the pair sum by

Input format

The first line contains 2 space-separated integers, n and k .
The second line contains n space-separated integers describing the values of array a.

Output Format

Print the number of pairs (i,j) where I<j and a[i]+a[j] is evenly divisible by k .

Constraint:
n = [2 to 100]
k = [1 to 100]
a[i] = [1 to 100]

Solution:

```
#!/bin/python3
import math
import os
import random
import re
import sys

# Complete the divisibleSumPairs function below.
def divisibleSumPairs(n, k, ar):
    sum_lst = list()
    res = list()
    comb_list = (list(combinations(ar, 2)))
    for i in comb_list:
        sum_lst.append(i[0] + i[1])
    for i in sum_lst:
        if (i % k) == 0:
            res.append(i)
    return (len(res))

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    nk = input().split()
    n = int(nk[0])
    k = int(nk[1])
    ar = list(map(int, input().rstrip().split()))
    result = divisibleSumPairs(n, k, ar)
    fptr.write(str(result) + '\n')
    fptr.close()
```

****

Activity 3:

Given a square matrix, calculate the absolute difference between the sums of its diagonals. The function must return an integer representing the absolute diagonal difference.
The diagonalDifference takes the following parameter:

arr: an array of integers.

Input Format
1. The first line contains a single integer,n , the number of rows and columns in the matrix arr.
2. Each of the next n lines describes a row,arr[i] , and consists of n space-separated integers arr[i][j].

Output Format
Print the absolute difference between the sums of the matrix's two diagonals as a single integer.

Constraint:
arr[i][j]={-100 to 100}

Solution:

```
#!/bin/python3

import math
import os
import random
import re
import sys

def diagonalDifference(arr):
    # Write your code here
    n = int(input().strip())
dSumLeft  = 0
dSumRight  = 0
for i in range(n):
    matrixRow = list(map(int,input().strip().split(' ')))
    dSumLeft += int(matrixRow[i])
    dSumRight += int(matrixRow[n-i-1])
print(abs(dSumLeft - dSumRight))
if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    n = int(input().strip())
    arr = []
    for _ in range(n):
        arr.append(list(map(int, input().rstrip().split())))
    result = diagonalDifference(arr)
    fptr.write(str(result) + '\n')
    fptr.close()
```

****

Activity 4:

You are given a number of sticks of varying lengths. You will iteratively cut the sticks into smaller sticks, discarding the shortest pieces until there are none left. At each iteration you will determine the length of the shortest stick remaining, cut that length from each of the longer sticks and then discard all the pieces of that shortest length. When all the remaining sticks are the same length, they cannot be shortened so discard them.

Given the lengths of n sticks, print the number of sticks that are left before each iteration until there are none left.

Input format:
The first line contains a single integer n, the size of arr.
The next line contains n space- separated integers, each an arr[i], where each value represents the length of the ith stick.

Output format:
For each operation, print the number of sticks that are present before the operation on separate lines.

Constraints:
n = [1 to 1000]
arr[i] = [1 to 1000]

Solution:

```
N= input()
num = map(int,raw_input().split())
val=[0]*1001
for i in num:
   val[i]+=1
counter = 0
val=val[::-1]
ans =[]
for i in val:
   if i>0:
      counter += i
      ans.append(counter)
ans = ans[::-1]
for i in ans:
   print i
```

****
Activity 5:

Given an array of integers, find and print the maximum number of integers you can select from the array such that the absolute difference between any two of the chosen integers is less than or equal to 1.

Input Format

The first line contains a single integer n, the size of the array arr.
The second line contains n space-separated integers.

Output Format

A single integer denoting the maximum number of integers you can choose from the array such that the absolute difference between any two of the chosen integers is <2.

Solution:

```
#!/bin/python3

import math
import os
import random
import re
import sys

def pickingNumbers(a):
    # Write your code here
    freq = [0 for i in range(101)]
    max_pair = 0
    for n in a:
        freq[n] += 1
        max_pair = max(max_pair, max((freq[n] + freq[n-1]), (freq[n] + freq[n+1])))
    return max_pair

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    n = int(input().strip())
    a = list(map(int, input().rstrip().split()))
    result = pickingNumbers(a)
    fptr.write(str(result) + '\n')
    fptr.close()
```

****