**Week9:BUILT-IN FUNCTIONS,USER-DEFINED FUNCTIONS,RECURSIVE FUNCTIONS**

1. Write a code to check whether product of digits at even places is divisible by sum of digits at odd place of a positive integer.

**Example Input:**

1256

**Output:**

TRUE

**Example Input:**

1595

**Output:**

FALSE

**PROGRAM:**

def productDigits(number):

number\_str=str(number)

product\_even=1

sum\_odd=0

for i,digit\_char in enumerate(number\_str):

digit=int(digit\_char)

if(i+1)%2==0:

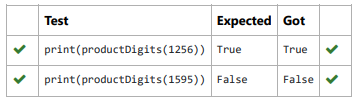
product\_even\*=digit

else:

sum\_odd+=digit

return product\_even%sum\_odd ==0

**OUTPUT:**

****

2. Given a number with maximum of 100 digits as input, find the difference between the sum

of odd and even position digits.

**Example input:**

1453

**Output:**

1

**PROGRAM:**

def differenceSum(n):

n=str(n)

l=[0,0]

for i in range(len(n)):

if i%2==0:

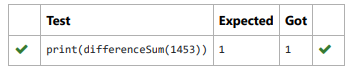
l[0]=l[0]+int(n[i])

else:

l[1]=l[1]+int(n[i])

return(abs(l[0]-l[1]))

**OUTPUT:**

****

3. A number is considered to be ugly if its only prime factors are 2, 3 or 5.

[1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, …] is the sequence of ugly numbers.

Task:

complete the function which takes a number n as input and checks if it's an ugly number.

return ugly if it is ugly, else return not ugly

Hint:

An ugly number U can be expressed as: U = 2^a \* 3^b \* 5^c, where a, b and c are nonnegative integers.

**PROGRAM:**

def checkUgly(n):

if n<=0:

return "not ugly"

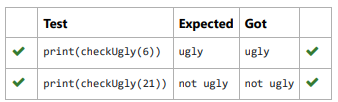
for p in [2,3,5]:

while n%p == 0:

n//=p

return "ugly" if n==1 else "not ugly"

**OUTPUT:**



4. complete function to implement coin change making problem i.e. finding the minimum

number of coins of certain denominations that add up to given amount of money.

The only available coins are of values 1, 2, 3, 4

**Example Input:**

16

**Output:**

4

**Example Input:**

25

**Output:**

7

**PROGRAM:**

def coinChange(target):

coins=[1,2,3,4]

dp=[float('inf')]\*(target +1)

dp[0]=0

for i in range(1,target+1):

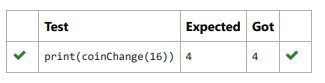
for coin in coins:

if coin <=i:

dp[i]=min(dp[i],dp[i-coin]+1)

return dp[target]

**OUTPUT:**

****

5. An abundant number is a number for which the sum of its proper divisors is greater than

the number itself. Proper divisors of the number are those that are strictly lesser than the number.

**Example input:**

12

**Output:**

Yes

**Example input:**

13

**Output:**

No

**PROGRAM:**

def abundant(number):

divisor\_sum =sum([divisor for divisor in range(1,number) i

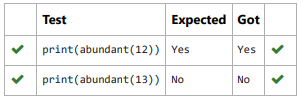
if divisor\_sum>number:

return "Yes"

else:

return "No"

**OUTPUT:**

****