9.Function:Built-in function, User defined Function, Recursive function

**Ex. No. : 9.1 Date:06/06/2024**

**Register No.: 231801047 Name: HANNAH JAMES**

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

Input Format:

Integer input from stdin.

Output Format:

return the minimum number of coins required to meet the given target.

Example Input:

16

Output:

4

Explanation:

We need only 4 coins of value 4 each

Example Input:

25

Output:

7

Explanation:

We need 6 coins of 4 value, and 1 coin of 1 value

PROGRAM:

def coinC (n):

coins=[1,2,3,4]

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

dp[0]=0

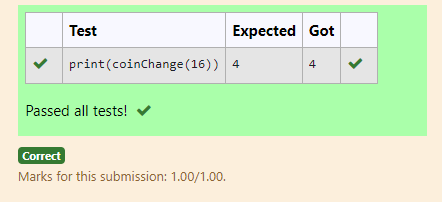
for i in range(1,n+1):

for coin in coins:

if coin<=i:

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

return dp[n]



**Ex. No. : 9.2 Date: 06/06/2024**

**Register No.: 231801047 Name: HANNAH JAMES**

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.

**For example:**

| **Test** | **Result** |
| --- | --- |
| print(checkUgly(6)) | ugly |
| print(checkUgly(21)) | not ugly |

PROGRAM:

def checkU (n):

if n <= 0:

return "not ugly"

while n % 2 == 0:

n //= 2

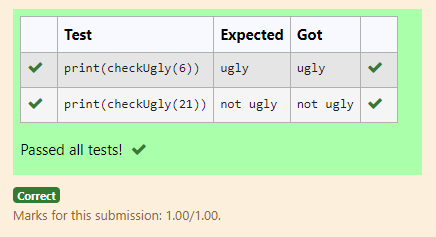
while n % 3 == 0:

n //= 3

while n % 5 == 0:

n //= 5

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



**Ex. No. : 9.3 Date: 06/06/2024**

**Register No.: 231801047 Name: HANNAH JAMES**

An automorphic number is a number whose square ends with the number itself.

For example, 5 is an automorphic number because 5\*5 =25. The last digit is 5 which same

as the given number.

If the number is not valid, it should display “Invalid input”.

If it is an automorphic number display “Automorphic” else display “Not Automorphic”.

Input Format:

Take a Integer from Stdin Output Format: Print Automorphic if given number is Automorphic number,otherwise Not Automorphic Example input: 5 Output: Automorphic Example input: 25 Output: Automorphic Example input: 7 Output: Not Automorphic

**For example:**

| **Test** | **Result** |
| --- | --- |
| print(automorphic(5)) | Automorphic |

PROGRAM:

def automorphic(n):

if(n<0):

return "Invalid input"

square = n \* n

n\_s=str(n)

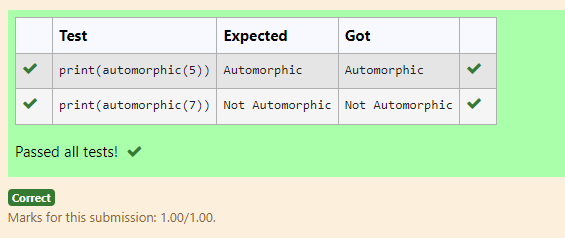
s\_s=str(square)

if s\_s.endswith(n\_s):

return "Automorphic"

else:

return "Not Automorphic"



**Ex. No. : 9.4 Date: 06/06/2024**

**Register No.: 231801047 Name: HANNAH JAMES**

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

of odd and even position digits.

Input Format:

Take a number in the form of String from stdin.

Output Format:

Print the difference between sum of even and odd digits

Example input:

1453

Output:

1

Explanation:

Here, sum of even digits is 4 + 3 = 7

sum of odd digits is 1 + 5 = 6.

Difference is 1.

Note that we are always taking absolute difference

PROGRAM:

def differenceS (number\_str):

number\_str=str(number\_str)

odd\_sum = 0

even\_sum = 0

for index, char in enumerate(number\_str):

digit = int(char)

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

even\_sum += digit

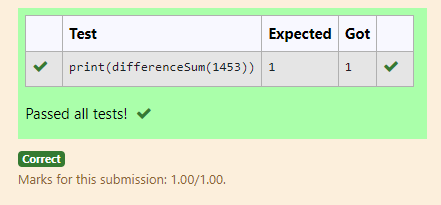
else:

odd\_sum += digit

difference = abs(even\_sum - odd\_sum)

return difference

print(differenceS (number\_str))



**Ex. No. : 9.5 Date: 06/06/2024**

**Register No.: 231801047 Name: HANNAH JAMES**

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.

Input Format:

Take input an integer from stdin

Output Format:

Return Yes if given number is Abundant. Otherwise, print No

Example input:

12

Output:

Yes

Explanation

The proper divisors of 12 are: 1, 2, 3, 4, 6, whose sum is 1 + 2 + 3 + 4 + 6 = 16. Since sum of

proper divisors is greater than the given number, 12 is an abundant number.

Example input:

13

Output:

No

Explanation

The proper divisors of 13 is: 1, whose sum is 1. Since sum of proper divisors is not greater

than the given number, 13 is not an abundant number.

**For example:**

| **Test** | **Result** |
| --- | --- |
| print(abundant(12)) | Yes |
| print(abundant(13)) | No |

PROGRAM:

def abundant(number):

d\_s=sum([divisor for divisor in range(1,number) if number % divisor == 0])

if d\_s>number:

return"Yes"

else:

return "No

