|  |  |  |  |
| --- | --- | --- | --- |
| **SVKM's-IOT, Dhule**Shri Vile Parle Kelavani Mandal's  **INSTITUTE OF TECHNOLOGY**  **DHULE (M.S.)**  **DEPARMENT OF COMPUTER ENGINEERING** | | | |
| **Subject :** Competitive Programmimg Lab | | | Remark |
| **Name : Jaykishan Natwar Varma** | | **Roll No. :** 68 |
| **Class :** TY. Comp. Engg. | **Batch :** T4 | **Division:** |
| **Expt. No. :** | **Date :** | | Signature |
| **Title :**  Counting | | |
|  | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **ASSIGNMENT/EXPERIMENT: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | |
| **Date of Performance:**  **Date of Submission:** | | |
| **Marks Split Up** | **Maximum Marks** | **Marks Obtained** |
| **Performance/Conduction** | **3** |  |
| **Report Writing** | **3** |  |
| **Attendance** | **2** |  |
| **Viva/Oral** | **2** |  |
| **Total Marks** | **10** |  |
| **Signature of Subject Teacher** | | |

**Title:** Counting

**Aim:** Gustavo knows how to count, but he is now learning how write numbers. As he is a very good student, he already learned 1, 2, 3 and 4. But he didn’t realize yet that 4 is different than 1, so he thinks that 4 is another way to write 1. Besides that, he is having fun with a little game he created himself: he make numbers (with those four digits) and sum their values. For instance: 132 = 1 + 3 + 2 = 6 112314 = 1 + 1 + 2 + 3 + 1 + 1 = 9 (remember that Gustavo thinks that 4 = 1) After making a lot of numbers in this way, Gustavo now wants to know how much numbers he can create such that their sum is a number n. For instance, for n = 2 he noticed that he can make 5 numbers: 11, 14, 41, 44 and 2 (he knows how to count them up, but he doesn’t know how to write five). However, he can’t figure it out for n greater than 2. So, he asked you to help him.

**Language used: Python**

**Platform Used: Pycharm, VS code etc.**

**Sample Input:** Input will consist on an arbitrary number of sets. Each set will consist on an integer n such that 1 ≤ n ≤ 1000. You must read until you reach the end of file.

**Sample Output**: For each number read, you must output another number (on a line alone) stating how much numbers Gustavo can make such that the sum of their digits is equal to the given number.

**Example:**

***Sample Input :***

2

3

***Sample Output:***

5

13

**Algorithm/Flowchart:**

1. **Precompute Valid Combinations**:
   * Create a function **count\_valid\_numbers(n)** that calculates the number of valid numbers whose digit sum is 𝑛*n*.
   * Use dynamic programming to compute **dp[x]**, which represents the number of valid combinations to achieve the digit sum 𝑥*x* using the digits {1, 1, 2, 3, 4}.
2. **Dynamic Programming Initialization**:
   * Initialize a DP array **dp** where **dp[0] = 1** (represents the empty set or a valid combination of digits that sum up to zero).
3. **Count Valid Combinations**:
   * Iterate over the digits {1, 2, 3, 4} (considering 4 as equivalent to 1).
   * For each digit 𝑑*d*, update the DP array **dp** to account for valid combinations by incrementing **dp[j]** where 𝑗*j* ranges from 𝑑*d* to 𝑛*n*.
4. **Handle Overlapping Digits**:
   * Since we have duplicate digits (two '1's), handle overlapping cases carefully to avoid double-counting combinations.
5. **Output Result**:
   * For each input 𝑛*n*, use the precomputed **dp** array to quickly determine the count of valid combinations for that particular sum.

**Code:**

from sys import stdin

def generate\_table(num):

# t[n] = 2\*t[n-1]+t[n-2]+t[n-3]

t = [0, 2, 5, 13]

for n in range(4, num+1):

t.append(2\*t[n-1]+t[n-2]+t[n-3])

return t

if \_\_name\_\_ == '\_\_main\_\_':

TABLE = generate\_table(1000)

while True:

line = stdin.readline()

if not line:

break

print(TABLE[int(line)])

**Input:-**

1

2

3

4

5

6

7

8

9

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

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

1000

**Output:-**

2

5

13

33

84

214

545

1388

3535

9003

22929

58396

148724

378773

964666

2456829

6257097

15935689

40585304

103363394

263247781

670444260

1707499695

4348691431

11075326817

28206844760

71837707768

182957587113

465959726754

1186714748389

3022346810645

7697368096433

19603797751900

49927310410878

127155786670089

323842681502956

824768460086879

2100535388346803

5349681918283441

13624667685000564

34699552676631372

88373454956546749

225071130274725434

573215268182628989

1459875121596530161

3718036641650414745

9469163673079988640

24116239109406922186

61419678533544247757

156424759849575406340

7804866167757385351726298167749579946964405850225254539132682472794981869745040537197592219996231328486687877730240352396489040560067523395940725030942516170568234738182127635234624655775531244438437118253542255365923486221253172456203933189283985689116139597563337647696143005496252287734941893682019406515104829885420261968884040123236083676226862353415881286645117793584639279853095668990201156175586714

**Conclusion:** In this way we implement The counting Problem using loops and conditional statements