

GOLDMAN SACHS CODING ROUND – 28th June 2021

1. Group Anagrams Together | Goldman Sachs

Write a program to input a set of words and group the anagrams together.

Sample inputs

Sample input-1

Enter the number of words : 6

Enter a word : bat

Enter a word : design

Enter a word : toc

Enter a word : signed

Enter a word : cot

Enter a word : tab

Sample output-1

The grouper anagrams are :

['bat', 'tab']

['design', 'signed']

['toc', 'cot']

Sample input-2

Enter the number of words : 8

Enter a word : beak

Enter a word : letter

Enter a word : bake

Enter a word : leg

Enter a word : yam

Enter a word : may

Enter a word : gel

Enter a word : eat

Sample output-2

The grouper anagrams are :

['beak', 'bake']

['letter'] ['leg', 'gel']

['yam', 'may']

['eat']

Code

```
def group_anagrams_together(words):
    arr = [''.join(sorted(word)) for word in words]
    dict = {}

    for i, e in enumerate(arr):
        dict.setdefault(e, []).append(i)

    for index in dict.values():
        print([words[i] for i in index])

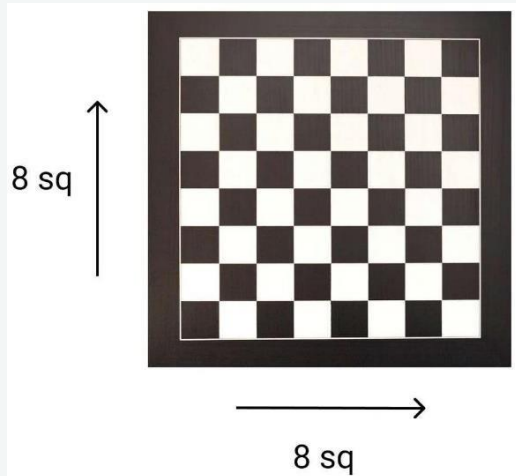
words = []

n=int(input("Enter the number of words : "))
for i in range(n):
    ele=input("Enter a word : ")
    words.append(ele)

print("The grouper anagrams are : ")
group_anagrams_together(words)
```

2. Number of Squares in a Chessboard | Goldman Sachs

Write program to return the possible number of squares in a 8*8 chessboard.



Explanation

The actual number of squares = $8 \times 8 = 64$.

But there are many more different sized squares.

Number of 1×1 squares = $8 \times 8 = 64$

Number of 2×2 squares = $7 \times 7 = 49$

Number of 3×3 squares = $6 \times 6 = 36$

Number of 4×4 squares = $5 \times 5 = 25$

Number of 5×5 squares = $4 \times 4 = 16$

Number of 6×6 squares = $3 \times 3 = 9$

Number of 7×7 squares = $2 \times 2 = 4$

Number of 8×8 squares = $1 \times 1 = 1$

Hence total number of square are

$64 + 49 + 36 + 25 + 16 + 9 = 204$

Code

```
1 def squares_in_chessboard(grid):
2     return (int((grid * (grid + 1) / 2)
3             * (2 * grid + 1) / 3) )
4
5 # Driver code
6 grid=8
7 print("Number of squares in an 8*8 chessboard : ", squares_in_chessboard(grid))
```

Output

Number of squares in an 8*8 chessboard : 204

3. Counting Sort | Goldman Sachs

Write a program to input an array of integers from the user and print the sorted array using counting sort.

Sample input-1

Enter the length of array : 3

Enter the element : 9

Enter the element : 0

Enter the element : 3

Sample output-1

Array sorted by counting sort is : [0, 3, 9]

Sample input-2

Enter the length of array :

6 Enter the element : 7

Enter the element : 3

Enter the element : 8

Enter the element : 1

Enter the element : 0

Enter the element : 2

Sample output-2

Array sorted by counting sort is : [0, 1, 2, 3, 7, 8]

Code

```
1 def counting_sort(arr):
2     result = [0] * 1
3
4     a = [0] * 10
5
6     for i in range(0, 1):
7         a[arr[i]] += 1
8
9     for i in range(1, 10):
10        a[i] += a[i - 1]
11
12    i = 1 - 1
13    while i >= 0:
14        result[a[arr[i]] - 1] = arr[i]
15        a[arr[i]] -= 1
16        i -= 1
17
```

Output

```
Enter the length of array : 6
Enter the element : 2
Enter the element : 0
Enter the element : 9
Enter the element : 5
Enter the element : 7
Enter the element : 1
Array sorted by counting sort is :
[0, 1, 2, 5, 7, 9]
```

4. Ugly Number | Goldman Sachs

Write a program to input an integer 'N' and return the N-th ugly number.

Explanation:

Ugly numbers are the numbers whose prime factors are 2, 3 or 5.

For example ugly numbers from 1 to 15, there are 11 ugly numbers 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15.

Sample inputs

Sample input-1

Enter the N-th value : 11

15

Sample input-2

Enter the N-th value : 40

144

Code

```
def ugly_number(n):
    a = [0] * n
    a[0] = 1
    i2 = i3 = i5 = 0
    multiple2 = 2
    multiple3 = 3
    multiple5 = 5
    for l in range(1, n):
        a[l] = min(multiple2, multiple3, multiple5)
        if a[l] == multiple2:
            i2 += 1
            multiple2 = a[i2] * 2

        if a[l] == multiple3:
            i3 += 1
            multiple3 = a[i3] * 3
```

```
        if a[l] == multiple5:
            i5 += 1
            multiple5 = a[i5] * 5

    return a[-1]
```

```
n = int(input("Enter the N-th value : "))
print(ugly_number(n))
```

Output

```
Enter the N-th value : 120
2700
```

5. Compute average of two numbers without overflow

Given two numbers, a and b. Compute the average of the two numbers.

The well know formula $(a + b) / 2$ may fail at the following case :

If, $a = b = (2^{31}) - 1$; i.e. INT_MAX.

Now, $(a+b)$ will cause overflow and hence formula $(a + b) / 2$ wont work

Code

```
INT_MAX=2147483647
```

```
# Function to compute average of two numbers
```

```
def compute_average(a,b):
```

```
    return (a // 2) + (b // 2) + ((a % 2 + b % 2) // 2)
```

```
# Driver code
```

```
if __name__ == "__main__":
```

```
    # Assigning maximum integer value
```

```
    a = INT_MAX
```

```
    b = INT_MAX
```

```
    # Average of two equal
```

```
    # numbers is the same number
```

```
    print( "Actual average : ",INT_MAX)
```

```
    # Function to get the
```

```
    # average of 2 numbers
```

```
    print( "Computed average : ", compute_average(a, b))
```