DS INTERNSHIP-PHYTHON FUNDAMENTALS

QUESTION 1

```
def generate_2d_array(X, Y):
    result = []
    for i in range(X):
        row = []
        for j in range(Y):
            row.append(i * j)
        result.append(row)
    return result

X, Y = map(int, input("Enter two digits (X,Y): ").split(","))
output_array = generate_2d_array(X, Y)
for row in output_array:
    print(row)

#OUTPUT:

Enter two digits (X,Y): 3,5
[0, 0, 0, 0, 0]
[0, 1, 2, 3, 4]
[0, 2, 4, 6, 8]
```

QUESTION *2*

```
input="without,hello,bag,world"
words=input.split(",")
words.sort()
result=",".join(words)
print(result)

#OUTPUT:

bag,hello,without,world
```

QUESTION 3

```
def remove_duplicates_and_sort(input_string):
  split=input_string.split()
  words=list(set(split))
  words.sort()
  result=" ".join(words)
  return result

input_string="hello world and practice makes perfect and hello world
again"
result=remove_duplicates_and_sort(input_string)
```

```
print(result)

#OUTPUT:

again and hello makes perfect practice world
```

QUESTION 4

```
for i in range(1000,3001):
  if(i%2==0):
    print(i,end=",")

#OUTPUT:

1000,1002,1004,1006,1008,1010,1012,1014,1016,1018,1020,1022,1024,1026,
1028,1030,1032,1034,1036,1038,1040,1042,1044,1046,1048,1050,1052,1054,
1056,1058,1060,1062,1064,1066,1068,1070,1072,1074,1076,1078,1080,1082,
1084,1086,1088,1090,1092,1094,1096,1098,1100,1102,1104,1106,1108,1110,
1112,1114,1116,1118,1120,1122,1124,1126,1128,1130,1132,1134,1136,1138,
1140,1142,1144,1146,1148,1150,1152,1154,1156,1158,1160,1162,1164,1166,
1168,1170,1172,1174,1176,1178,1180,1182,1184,1186,1188,1190,1192,1194,
1196,1198,1200,1202,1204,1206,1208,1210,1212,1214,1216,1218,1220,1222,
1224,1226,1228,1230,1232,1234,1236,1238,1240,1242,1244,1246,1248,1250,
1252,1254,1256,1258,1260,1262,1264,1266,1268,1270,1272,1274,1276,1278,
1280,1282,1284,1286,1288,1290,1292,1294,1296,1298,1300,1302,1304,1306,
1308,1310,1312,1314,1316,1318,1320,1322,1324,1326,1328,1330,1332,1334,
1336,1338,1340,1342,1344,1346,1348,1350,1352,1354,1356,1358,1360,1362,
1364,1366,1368,1370,1372,1374,1376,1378,1380,1382,1384,1386,1388,1390,
1392,1394,1396,1398,1400,1402,1404,1406,1408,1410,1412,1414,1416,1418,
1420,1422,1424,1426,1428,1430,1432,1434,1436,1438,1440,1442,1444,1446,
1448,1450,1452,1454,1456,1458,1460,1462,1464,1466,1468,1470,1472,1474,
1476,1478,1480,1482,1484,1486,1488,1490,1492,1494,1496,1498,1500,1502,
1504,1506,1508,1510,1512,1514,1516,1518,1520,1522,1524,1526,1528,1530,
1532,1534,1536,1538,1540,1542,1544,1546,1548,1550,1552,1554,1556,1558,
1560,1562,1564,1566,1568,1570,1572,1574,1576,1578,1580,1582,1584,1586,
1588,1590,1592,1594,1596,1598,1600,1602,1604,1606,1608,1610,1612,1614,
1616,1618,1620,1622,1624,1626,1628,1630,1632,1634,1636,1638,1640,1642,
1644,1646,1648,1650,1652,1654,1656,1658,1660,1662,1664,1666,1668,1670,
1672,1674,1676,1678,1680,1682,1684,1686,1688,1690,1692,1694,1696,1698,
1700,1702,1704,1706,1708,1710,1712,1714,1716,1718,1720,1722,1724,1726,
1728,1730,1732,1734,1736,1738,1740,1742,1744,1746,1748,1750,1752,1754,
1756,1758,1760,1762,1764,1766,1768,1770,1772,1774,1776,1778,1780,1782,
1784,1786,1788,1790,1792,1794,1796,1798,1800,1802,1804,1806,1808,1810,
1812,1814,1816,1818,1820,1822,1824,1826,1828,1830,1832,1834,1836,1838,
1840,1842,1844,1846,1848,1850,1852,1854,1856,1858,1860,1862,1864,1866,
1868,1870,1872,1874,1876,1878,1880,1882,1884,1886,1888,1890,1892,1894,
1896,1898,1900,1902,1904,1906,1908,1910,1912,1914,1916,1918,1920,1922,
1924,1926,1928,1930,1932,1934,1936,1938,1940,1942,1944,1946,1948,1950,
1952,1954,1956,1958,1960,1962,1964,1966,1968,1970,1972,1974,1976,1978,
1980,1982,1984,1986,1988,1990,1992,1994,1996,1998,2000,2002,2004,2006,
```

```
2008,2010,2012,2014,2016,2018,2020,2022,2024,2026,2028,2030,2032,2034,
2036,2038,2040,2042,2044,2046,2048,2050,2052,2054,2056,2058,2060,2062,
2064,2066,2068,2070,2072,2074,2076,2078,2080,2082,2084,2086,2088,2090,
2092,2094,2096,2098,2100,2102,2104,2106,2108,2110,2112,2114,2116,2118,
2120,2122,2124,2126,2128,2130,2132,2134,2136,2138,2140,2142,2144,2146,
2148,2150,2152,2154,2156,2158,2160,2162,2164,2166,2168,2170,2172,2174,
2176,2178,2180,2182,2184,2186,2188,2190,2192,2194,2196,2198,2200,2202,
2204,2206,2208,2210,2212,2214,2216,2218,2220,2222,2224,2226,2228,2230,
2232,2234,2236,2238,2240,2242,2244,2246,2248,2250,2252,2254,2256,2258,
2260,2262,2264,2266,2268,2270,2272,2274,2276,2278,2280,2282,2284,2286,
2288,2290,2292,2294,2296,2298,2300,2302,2304,2306,2308,2310,2312,2314,
2316,2318,2320,2322,2324,2326,2328,2330,2332,2334,2336,2338,2340,2342,
2344,2346,2348,2350,2352,2354,2356,2358,2360,2362,2364,2366,2368,2370,
2372,2374,2376,2378,2380,2382,2384,2386,2388,2390,2392,2394,2396,2398,
2400,2402,2404,2406,2408,2410,2412,2414,2416,2418,2420,2422,2424,2426,
2428,2430,2432,2434,2436,2438,2440,2442,2444,2446,2448,2450,2452,2454,
2456,2458,2460,2462,2464,2466,2468,2470,2472,2474,2476,2478,2480,2482,
2484,2486,2488,2490,2492,2494,2496,2498,2500,2502,2504,2506,2508,2510,
2512,2514,2516,2518,2520,2522,2524,2526,2528,2530,2532,2534,2536,2538,
2540,2542,2544,2546,2548,2550,2552,2554,2556,2558,2560,2562,2564,2566,
2568,2570,2572,2574,2576,2578,2580,2582,2584,2586,2588,2590,2592,2594,
2596,2598,2600,2602,2604,2606,2608,2610,2612,2614,2616,2618,2620,2622,
2624,2626,2628,2630,2632,2634,2636,2638,2640,2642,2644,2646,2648,2650,
2652,2654,2656,2658,2660,2662,2664,2666,2668,2670,2672,2674,2676,2678,
2680,2682,2684,2686,2688,2690,2692,2694,2696,2698,2700,2702,2704,2706,
2708,2710,2712,2714,2716,2718,2720,2722,2724,2726,2728,2730,2732,2734,
2736,2738,2740,2742,2744,2746,2748,2750,2752,2754,2756,2758,2760,2762,
2764,2766,2768,2770,2772,2774,2776,2778,2780,2782,2784,2786,2788,2790,
2792,2794,2796,2798,2800,2802,2804,2806,2808,2810,2812,2814,2816,2818,
2820,2822,2824,2826,2828,2830,2832,2834,2836,2838,2840,2842,2844,2846,
2848,2850,2852,2854,2856,2858,2860,2862,2864,2866,2868,2870,2872,2874,
2876,2878,2880,2882,2884,2886,2888,2890,2892,2894,2896,2898,2900,2902,
2904,2906,2908,2910,2912,2914,2916,2918,2920,2922,2924,2926,2928,2930,
2932,2934,2936,2938,2940,2942,2944,2946,2948,2950,2952,2954,2956,2958,
2960,2962,2964,2966,2968,2970,2972,2974,2976,2978,2980,2982,2984,2986,
2988,2990,2992,2994,2996,2998,3000,
```

QUESTION 5

```
def sentence(input_string):
    letter_count=0
    digits_count=0


    for char in input_string:
        if char.isalnum():
            if char.islower():
                letter_count+=1
            else:
```

```
                digits_count+=1

    print("LETTERS:{}".format(letter_count))
    print("DIGITS:{}".format(digits_count))


input_string="hello world! 123"
sentence(input_string)

#OUTPUT:

LETTERS:10
DIGITS:3
```

QUESTION 6

```
def sentence(input_letters):
  uppercase_count=0
  lowercase_count=0
  for char in input_letters:
    if char .isupper():
      uppercase_count+=1
    elif char .islower():
      lowercase_count+=1
  print(f"UPPER CASE:{uppercase_count}")
  print(f"LOWER CASE:{lowercase_count}")

input_letters="Hello world!"
sentence(input_letters)

#OUTPUT:

UPPER CASE:1
LOWER CASE:9
```

QUESTION 7

```
def compute_net_amount(transactions):
    net_amount=0
    for transaction in transactions:
        type,amount=transaction.split()
        amount=int(amount)
        if type=="D":
            net_amount +=amount
        elif type=="W":
            net_amount -=amount
    print("Net amount is",net_amount)

transactions=input()
output=transactions.split(",")
```

```
compute_net_amount(output)

#OUTPUT:

D 300,D 300,W 200, D 100
Net amount is 500
```

QUESTION 8

```
def valid_password(password):
    has_special_char=False
    words=""
    for char in password:
        if char in "abcdefghijklmnopqrstuvwxyz":
             words+=char
        elif char in "0123456789":
            words+=char
        elif char in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
            words+=char
        elif char in "$#@":
            words+=char
            has_special_char=True
    if len(words)==6 or len(words)<=12 and has_special_char:
        print(words)
    else:
        print("invalid")

password=input("Input is:")
valid_password(password)

#OUTPUT:

Input is:A,B,d,1,2,3,4,@,1
ABd1234@1
```

QUESTION 9

```
def sorting(input):
   input.sort(key=lambda x: (x[0], int(x[1]), int(x[2])))
   print(list(input))

input=(
    ("Tom", "19", "80"),
    ("John", "20", "90"),
    ("Jony", "17", "91"),
    ("Jony", "17", "93"),
    ("Json", "21", "85")
)
output=list(input)
```

```
sorting(output)

#OUTPUT:

[('John', '20', '90'), ('Jony', '17', '91'), ('Jony', '17', '93'),
('Json', '21', '85'), ('Tom', '19', '80')]
```

QUESTION 10

```
import math
def compute_distance(movements):
  x,y=0,0
  for movement in movements:
    directions, steps=movement.split()
    steps=int(steps)
    if directions=="UP":
      y+=steps
    elif directions=="DOWN":
        y-=steps
    elif directions=="LEFT":
      x-=steps
    elif directions=="RIGHT":
      x+=steps

  distance=math.sqrt(x**2+y**2)
  rounding_value= round(distance)
  print("Distance",rounding_value)


movements=input().split(",")
compute_distance(movements)

#OUTPUT:

UP 5,DOWN 3,LEFT 3,RIGHT 2
Distance 2
```

QUESTION 11

```
def count_continuous_occurrences(string):
    result = ""
    count = 1

    for i in range(1, len(string)):
        if string[i] == string[i - 1]:
            count += 1
        else:
            result += string[i - 1] + str(count)
            count = 1
```

```
    result += string[-1] + str(count)  # Add the last character and
its count
    print(result)


input_string = "Aabbcdeefffaabbcc"
count_continuous_occurrences(input_string.lower())

#OUTPUT:

a2b2c1d1e2f3a2b2c2
```

QUESTION 12

```
def find_pairs_with_sum_9(alphanumeric_str):
    result = []
    current_sum = 0
    prev_alpha = None

    for char in alphanumeric_str:
        if char.isalpha():
            if prev_alpha is not None and current_sum == 9:
                result.append((prev_alpha, char))
            prev_alpha = char
            current_sum = 0
        elif char.isdigit():
            current_sum += int(char)

    return result

# Example usage:
input_str1 = "a54b12c"
output1 = find_pairs_with_sum_9(input_str1)
for pair in output1:
  join1=(", ".join([f"{pair[0]}{pair[1]}"]))
  print(",".join(join1))
print("-" * 8)


input_str2 = "a55b234cd9f63de54x3m"
output2 = find_pairs_with_sum_9(input_str2)
for pair in output2:
  join2=", ".join([f"{pair[0]}{pair[1]}"])
  print(",".join(join2))

#OUTPUT:

a,b
--------
b,c
```

```
d,f
f,d
e,x
```

QUESTION 13

```
def count_pairs(binary_string):
    pair_count = 0

    num_ones = binary_string.count('1')

    if num_ones >= 2:
        pair_count = num_ones * (num_ones - 1) // 2

    return pair_count
input1 = "100101"
input2 = "1001101010010"

output1 = count_pairs(input1)
output2 = count_pairs(input2)

print(output1)
print(output2)


#OUTPUT:

3
15
```

QUESTION 14

```
def find_minimum_denominations(valid_currency, money):
    valid_currency.sort(reverse=True)
    denominations = {}
    for coin in valid_currency:
        if money >= coin:
            count = money // coin
            denominations[coin] = count
            money -= count * coin
    return denominations

valid_currency1 = [1, 2, 5, 10, 20, 50, 100, 200, 500, 2000]
money1 = 210
print("Expected Output 1:")
for coin, count in find_minimum_denominations(valid_currency1,
money1).items():
    print(f"{coin}-{count}")
print("-" *20)
```

```
valid_currency2 = [1, 2, 5, 10, 20, 50, 100, 200, 500]
money2 = 556
print("\nExpected Output 2:")
for coin, count in find_minimum_denominations(valid_currency2,
money2).items():
    print(f"{coin}-{count}")

print("-" *20)

valid_currency3 = [1, 2, 5, 10, 20, 50, 100, 200, 500, 2000]
money3 = 2000
print("\nExpected Output 3:")
for coin, count in find_minimum_denominations(valid_currency3,
money3).items():
    print(f"{coin}-{count}")

print("-" *20)

valid_currency4 = [1, 2, 5, 10, 20, 50, 100, 500, 1000]
money4 = 210
print("\nExpected Output 4:")
for coin, count in find_minimum_denominations(valid_currency4,
money4).items():
    print(f"{coin}-{count}")

print("-" *20)

valid_currency5 = [1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]
money5 = 2000
print("\nExpected Output 5:")
for coin, count in find_minimum_denominations(valid_currency5,
money5).items():
    print(f"{coin}-{count}")

#OUTPUT:

Expected Output 1:
200-1
10-1
--------------------

Expected Output 2:
500-1
50-1
5-1
1-1
--------------------

Expected Output 3:
2000-1
```

```
--------------------

Expected Output 4:
100-2
10-1
--------------------

Expected Output 5:
1000-2
```

QUESTION 15

```
import math

def choose_non_consecutive_stops(n, m):
    return math.comb(n - m + 1, m)

n1, m1 = 12, 4
output1 = choose_non_consecutive_stops(n1, m1)
print(f"{output1}")

print("-" * 20)

n2, m2 = 16, 5
output2 = choose_non_consecutive_stops(n2, m2)
print(f"{output2}")


#OUTPUT:

126
--------------------
792
```

QUESTION 16

```
def play_game():
    player_a_score = 0
    player_b_score = 0

    while player_a_score < 5 and player_b_score < 5:
        player_a_choice = input("Player A, choose
(Stone/Paper/Scissor): ").lower()
        player_b_choice = input("Player B, choose
(Stone/Paper/Scissor): ").lower()

        if player_a_choice == player_b_choice:
            print("DRAW")
        elif (player_a_choice == "stone" and player_b_choice ==
"scissor") or \
```

```
            (player_a_choice == "paper" and player_b_choice ==
"stone") or \
            (player_a_choice == "scissor" and player_b_choice ==
"paper"):
            print("Player A wins")
            player_a_score += 1
        else:
            print("Player B wins")
            player_b_score += 1

    print("\nFinal Scores:")
    print("Player A:", player_a_score)
    print("Player B:", player_b_score)

play_game()


#OUTPUT:

Player A, choose (Stone/Paper/Scissor): stone
Player B, choose (Stone/Paper/Scissor): stone
DRAW
Player A, choose (Stone/Paper/Scissor): stone
Player B, choose (Stone/Paper/Scissor): paper
Player B wins
Player A, choose (Stone/Paper/Scissor): stone
Player B, choose (Stone/Paper/Scissor): scissor
Player A wins
Player A, choose (Stone/Paper/Scissor): paper
Player B, choose (Stone/Paper/Scissor): stone
Player A wins
Player A, choose (Stone/Paper/Scissor): paper
Player B, choose (Stone/Paper/Scissor): paper
DRAW
Player A, choose (Stone/Paper/Scissor): paper
Player B, choose (Stone/Paper/Scissor): scissor
Player B wins
Player A, choose (Stone/Paper/Scissor): scissor
Player B, choose (Stone/Paper/Scissor): scissor
DRAW
Player A, choose (Stone/Paper/Scissor): scissor
Player B, choose (Stone/Paper/Scissor): stone
Player B wins
Player A, choose (Stone/Paper/Scissor): scissor
Player B, choose (Stone/Paper/Scissor): paper
Player A wins
Player A, choose (Stone/Paper/Scissor): paper
Player B, choose (Stone/Paper/Scissor): stone
Player A wins
Player A, choose (Stone/Paper/Scissor): scissor
```

```
Player B, choose (Stone/Paper/Scissor): paper
Player A wins

Final Scores:
Player A: 5
Player B: 3
```

QUESTION 17

```python
def validate_email(email):
    if email.count('@') != 1:
        return False

    local_part, domain_part = email.split('@')
    allowed_chars = set("abcdefghijklmnopqrstuvwxyz0123456789._")
    for char in local_part:
        if char not in allowed_chars:
            return False
    for char in domain_part.lower():
        if char not in allowed_chars:
            return False

    return True

email1 = "user@example.com"
email2 = "User123@domain.com"

print(validate_email(email1))
print(validate_email(email2))


#OUTPUT:

True
False
```

QUESTION *18*

```python
#Pattern:1

def pattern(rows):
    num = 1
    for i in range(1, rows + 1):
        for j in range(i):
            print(num, end=" ")
            num += 1
        print()

pattern(4)
```

```
#OUTPUT:

1
2 3
4 5 6
7 8 9 10

#Pattern 2

def print_diamond(rows):
    for i in range(1, rows + 1):
        print(" " * (rows - i), end="")
        for j in range(i):
            print("*", end=" ")
        print()

    for i in range(rows - 1, 0, -1):
        print(" " * (rows - i), end="")
        for j in range(i):
            print("*", end=" ")
        print()


rows = int(input("Enter the number of rows: "))
print_diamond(rows)

#OUTPUT:

Enter the number of rows: 4
   *
  * *
 * * *
* * * *
 * * *
  * *
   *

#Pattern 3
def pattern(rows):
    num = 1
    for i in range(1, rows + 1):
        for j in range(i):
            print(num, end=" ")
            num += 1
        print()
    for i in range(rows - 1, 0, -1):
        for j in range(i):
            print(num, end=" ")
            num += 1
```

```
        print()


pattern(4)

#OUTPUT:

1
2 3
4 5 6
7 8 9 10
11 12 13
14 15
16

#Pattern 4

def print_pattern(rows):
    for i in range(rows):
        if i == 0:
            print(" " * 2 + "***")
        elif i == rows - 1:
            print(" " * 2 + "* * *")
        elif i == rows // 2:
            print("*" * 3 + " " * (rows - 4) + "*")
        else:
            print("*")

rows = 7
print_pattern(rows)

#OUTPUT:

  ***
*
*
***   *
*
*
  * * *



#pattern 5

def pattern(rows):
    for i in range(rows):
        for j in range(rows):
            if i == 0 or i == rows - 1 or j == 0 or j == rows - 1:
                print("1", end=" ")
            else:
```

```
                print("0", end=" ")
        print()


pattern(5)


#OUTPUT:
```

```
1 1 1 1 1
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1
1 1 1 1 1
```

QUESTION 19

```
def cyclic_rotate_string(s, direction, times):
    n = len(s)
    rotated_s = s

    if direction == 1:
        for _ in range(times):
            rotated_s = rotated_s[1:] + rotated_s[0]
            print(rotated_s)
    elif direction == 2:
        for _ in range(times):
            rotated_s = rotated_s[-1] + rotated_s[0:4:1]
            print(rotated_s)
    else:
        print("Invalid direction. Please choose 1 or 2.")

    return rotated_s

input_str1 = "happy"
direction1 = 1
times1 = 2
output1 = cyclic_rotate_string(input_str1, direction1, times1)
print("Output for input 1:", output1)


input_str2 = "happy"
direction2 = 2
times2 = 2
output2= cyclic_rotate_string(input_str2, direction2, times2)
print("Output for input 2:", output2)

#OUTPUT:
```

```
appyh
ppyha
Output for input 1: ppyha
yhapp
pyhap
Output for input 2: pyhap
```

QUESTION 20

```
def get_user_input(component_name):
    try:
        return float(input(f"Enter {component_name}: "))
    except ValueError:
        print(f"Invalid input for {component_name}. Please enter a
valid numeric value.")
        return get_user_input(component_name)

healthy_patient_data = {
    "Sugar level": 15,
    "Blood pressure": 32,
    "Heartbeat rate": 71,
    "Weight": 65,
    "Fat percentage": 10
}
dict={}

user_data = {}
for component in healthy_patient_data:
    user_data[component] = get_user_input(component)

for component, value in user_data.items():
    diff =healthy_patient_data[component] - value
    dict[component]=diff
print(dict)

for component, value in user_data.items():
    diff =healthy_patient_data[component] - value
    print(f"{component}: {diff}")

    if diff > 0:
        print(f"{component} is {diff} more than the ideal value")
    elif diff < 0:
        print(f"{component} is {diff} less than the ideal value")
    else:
        print(f"{component} is {diff} at the ideal value")


#OUTPUT:
```

```
Enter Sugar level: 56
Enter Blood pressure: 120
Enter Heartbeat rate: 45
Enter Weight: 67
Enter Fat percentage: 67
{'Sugar level': -41.0, 'Blood pressure': -88.0, 'Heartbeat rate':
26.0, 'Weight': -2.0, 'Fat percentage': -57.0}
Sugar level: -41.0
Sugar level is -41.0 less than the ideal value
Blood pressure: -88.0
Blood pressure is -88.0 less than the ideal value
Heartbeat rate: 26.0
Heartbeat rate is 26.0 more than the ideal value
Weight: -2.0
Weight is -2.0 less than the ideal value
Fat percentage: -57.0
Fat percentage is -57.0 less than the ideal value
```

QUESTION 21

```python
def is_armstrong_number(num):
    num_str = str(num)
    num_digits = len(num_str)
    armstrong_sum = sum(int(digit) ** num_digits for digit in num_str)
    return armstrong_sum == num

input_number = 1634
if is_armstrong_number(input_number):
    print("Armstrong number")
else:
    print("Not an Armstrong number")


#OUTPUT:

Armstrong number
```

QUESTION 22

```python
def decimal_to_binary(num):
    binary_str = ""
    while num > 0:
        binary_str = str(num % 2) + binary_str
        num //= 2
    return binary_str

input_decimal1 = 12
input_decimal2 = 20
print(f"{input_decimal1} in binary:
```

```
{decimal_to_binary(input_decimal1)}")
print(f"{input_decimal2} in binary:
{decimal_to_binary(input_decimal2)}")


#OUTPUT:

12 in binary: 1100
20 in binary: 10100
```

QUESTION 23

```
def is_perfect_number(num):
    divisors_sum = 0
    for divisor in range(1, num):
        if num % divisor == 0:
            divisors_sum += divisor
    return divisors_sum == num

input_num = 28
if is_perfect_number(input_num):
    print(f"{input_num} is a perfect number")
else:
    print(f"{input_num} is not a perfect number")


#OUTPUT:

28 is a perfect number
```