0_toolcase

```
import sys
import time
import math
import re
import collections
import bisect

# ================ # kOj
def kOj(listy):
    return ''.join(listy)

# ================ # kpj
def kpj(listy):
    return ' '.join(listy)

# ================ # kcj
def kcj(listy):
    return ','.join(listy)

# ================ # klnj
def klnj(listy):
    return '\n'.join(listy)

# ================ # k_j_awith
def k_j_awith(listy, tara):
    return f'{tara}'.join(listy)

# ================ # matrix_out_print (kpj needed)
def matrix_out_print(amatrix):
    for arow in amatrix:
        print(kpj(arow))

# ================ # matrix_otto_print (kOj needed)
def matrix_otto_print(amatrix):
    for arow in amatrix:
        print(kOj(arow))

# ================ # cartland
def cartland(n0):
    if n0 == 0:
        c = 1
    else:
        c = 1
        for i0 in range(n0):
            c = c * (4 * i0 + 2) // (i0 + 2)
    return c

# ================ # fry_chicken
def fry_chicken(listy, k):
    listy.sort(reverse=True)
    res0 = sum(listy) / k
    if listy[0] <= res0:
        return res0
    else:
        new_listy = listy[1:]
        nk = k - 1
        return fry_chicken(new_listy, nk)

# ================ # mov_hanoi
def mov_hanoi(n0, a0, b0):
    return f'{n0}:{a0}->{b0}'

# ================ # solv_hanoi (mov_hanoi needed)
def solv_hanoi(n0, a0, b0, c0):
    if n0 == 1:
        return [mov_hanoi(n0, a0, c0)]
    else:
        jit_sp_solv_hanoi = []
        for i0 in solv_hanoi(n0 - 1, a0, c0, b0):
            jit_sp_solv_hanoi.append(str(i0))
        jit_sp_solv_hanoi.append(mov_hanoi(n0, a0, c0))
        for i0 in solv_hanoi(n0 - 1, b0, a0, c0):
            jit_sp_solv_hanoi.append(str(i0))
        return jit_sp_solv_hanoi

# ================ # protect_matrix
def protect_matrix(amatrix):
    protected_matrix = [['#' for mati in range(len(amatrix[0]) + 2)]]
    for arow in amatrix:
        protected_row = ['#']
        for matj in range(len(amatrix[0])):
            protected_row.append(arow[matj])
        protected_row.append('#')
        protected_matrix.append(protected_row)
    protected_matrix.append(['#' for mati in range(len(amatrix[0]) + 2)])
    return protected_matrix

# ================ # protect_matrixO
def protect_matrixO(amatrix):
    protected_matrix = [['O' for mati in range(len(amatrix[0]) + 2)]]
    for arow in amatrix:
        protected_row = ['O']
        for matj in range(len(amatrix[0])):
            protected_row.append(arow[matj])
        protected_row.append('O')
        protected_matrix.append(protected_row)
    protected_matrix.append(['O' for mati in range(len(amatrix[0]) + 2)])
    return protected_matrix

# ================ # protect_matrix_awith
def protect_matrix_awith(amatrix, withto_a):
    protected_matrix = [[withto_a for mati in range(len(amatrix[0]) + 2)]]
    for arow in amatrix:
        protected_row = [withto_a]
        for matj in range(len(amatrix[0])):
            protected_row.append(arow[matj])
        protected_row.append(withto_a)
        protected_matrix.append(protected_row)
    protected_matrix.append([withto_a for mati in range(len(amatrix[0]) + 2)])
    return protected_matrix

# ================ # mount
def mount(listie):
    mount_altitude = [0]
    for i in range(1, len(listie)):
        if listie[i] > listie[i - 1]:
            mount_altitude.append(mount_altitude[-1] + 1)
        elif listie[i] < listie[i - 1]:
            mount_altitude.append(mount_altitude[-1] - 1)
        else:
            mount_altitude.append(mount_altitude[-1])
    base_altitude = min(mount_altitude) - 1
    mount_waterfill = -len(listie) * base_altitude
    mount_seasum = sum(mount_altitude)
    mount_landsum = mount_seasum + mount_waterfill
    return mount_landsum

# ================ # up
def up(sest_001):
    blacklist_001 = 0
    if len(sest_001) == 1:
        return True
    else:
        for _ in range(len(sest_001) - 1):
            if sest_001[_] > sest_001[_ + 1]:
                blacklist_001 += 1
        if blacklist_001 == 0:
            return True
        else:
            return False

# ================ # poup
def poup(sest_002):
    while not up(sest_002):
        for _ in range(len(sest_002) - 1):
            if sest_002[_] > sest_002[_ + 1]:
                sest_002[_], sest_002[_ + 1] = sest_002[_ + 1], sest_002[_]
    return sest_002

# ================ # permute0 (所有排列方式)
def permute0(listy):
    if len(listy) == 1:
        return [[listy[0]]]
    else:
        al_listy = []
        for j_in_listy in listy:
            n_listy = [j_in_listy]
            listy0 = listy[:]
            listy0.remove(j_in_listy)
            for k_in_listyf in permute0(listy0):
                for k_in_listy in k_in_listyf:
                    n_listy.append(k_in_listy)
                al_listy.append(n_listy)
                n_listy = [j_in_listy]
        return al_listy

# ================ # subsets (所有子集)
def subsets(listy):
    res0 = [[]]
    for i0 in listy:
        res0 += [r0 + [i0] for r0 in res0]
    return res0

# ================ # is_prime
def is_prime(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

dir4 = [[0, 1], [0, -1], [1, 0], [-1, 0]]
dir8 = [[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 1], [1, -1], [1, 0], [1, 1]]
```