**ali\_test1.py**

def addSum(n, target):

if nums[n] == target:

flag = True

elif n == 1:

return

else:

addSum(n-1, target-nums[n])

addSum(n-1, target)

if \_\_name\_\_ == "\_\_main\_\_":

nums = [1,2,3,4,5,6]

n = len(nums) -1

target = 12

flag = False

addSum(n, target)

if flag:

print('yes')

else:

print('no')

**ali\_test.py**

# coding=gbk

# 数组中任意几个元素的和是否等于m

def addSum(nums, target):

nums = sorted(nums)

loc = 0

init\_loc = 0

sum = nums[loc]

stack = []

stack.append([nums[loc], loc]);

loc += 1

while sum != target:

if loc < len(nums) and sum + nums[loc] <= target:

sum += nums[loc]

stack.append([nums[loc], loc])

loc += 1

else:

if stack:

v = stack.pop()

sum -= v[0]

loc = v[1] + 1

else:

sum = 0;

init\_loc += 1

loc = init\_loc

res = []

while stack:

res.append(stack.pop()[0])

return res

nums = [1,2,3,4,5,6]

target = 13

res = addSum(nums, target)

print(res)

**huawei2015\_1.py**

# coding = gbk

# 按要求分解字符串，输入两个数M，N；M代表输入的M串字符串，N代表输出的每串字符串的位数，

# 不够补0。例如：输入2,8， “abc” ，“123456789”，则输出为“abc00000”,“12345678“,”90000000”

alist = list(input().split(','))

# print(m,n)

# slist = list(input().split(','))

# print(alist)

# print(alist)

def output\_str(a\_str,n):

if len(a\_str) <= n:

while len(a\_str) < n:

a\_str += '0'

# slist.append(a\_str)

# return a\_str

print(a\_str)

else:

print(a\_str[:n])

a\_str = a\_str[n:]

output\_str(a\_str,n)

m = int(alist[0])

n = int(alist[1])

alist = alist[2:]

for i in range(m):

output\_str(alist[i],n)

# xlist += slist

# print(xlist)

**Huawei2015\_2.py**

# coding = gbk

# 第二题：去除重复字符并排序

# 运行时间限制：无限制

# 内容限制： 无限制

# 输入： 字符串

# 输出： 去除重复字符并排序的字符串

# 样例输入： aabcdefff

# 样例输出： abcdef

# s = input()

# x = ''.join(sorted(set(s)))

# print(x)

s = input()

x = ''

for i in s:

if i not in x:

x += i

print(x)

def sort\_str(s):

s = list(s)

for i in range(len(s)):

for j in range(i,len(s)-1):

if s[j] > s[j+1]:

temp = s[j]

s[j] = s[j+1]

s[j+1] = temp

x = ''.join(s)

return x

print(sort\_str(x))

**huawei2015\_3.py**

# coding = gbk

# 第三题：等式变换

# 输入一个正整数X，在下面的等式左边的数字之间添加+号或者-号，使得等式成立。

# 1 2 3 4 5 6 7 8 9 = X

# 比如：

# 12-34+5-67+89 = 5

# 1+23+4-5+6-7-8-9 = 5

# 请编写程序，统计满足输入整数的所有整数个数。

# 输入： 正整数，等式右边的数字

# 输出： 使该等式成立的个数

# 样例输入：5

# 样例输出：21

# //动态规划

# //动态方程（有点难理解）：当前种类=符号位加号+符号为减号+没有符号的种类

# //dp(before,des,n,ex)= dp(before - 1, before, res + des,1) + dp(before - 1, before, res - des,1) + dp(before - 1, before\*pow(10, ex)+des, res,ex+1);

# // before: 需要判定的符号前面的数字的个数，初始为8

# // des: 需要判定的符号后面的数字，初始为9

# // n:方程右边的结果

# // ex:阶乘数，因为符号有三种可能，加号，减号，或者没有，如果没有，那么ex就用于计算当前值

x = int(input())

def count\_x(before, des, res, ex):

if before == 0:

if des == res:

return 1

else:

return 0

else:

return count\_x(before-1, before, res+des, 1) + count\_x(before-1, before, res-des,1) + count\_x(before-1, before\*10\*\*ex+des, res, ex+1)

print(count\_x(8,9,x,1))

**huawei2016\_1.py**

# coding = gbk

# 输入包括多组测试数据。

# 每组输入第一行是两个正整数N和M（0 < N <= 30000,0 < M < 5000）,分别代表学生的数目和操作的数目。

# 学生ID编号从1编到N。

# 第二行包含N个整数，代表这N个学生的初始成绩，其中第i个数代表ID为i的学生的成绩

# 接下来又M行，每一行有一个字符C（只取‘Q’或‘U’），和两个正整数A,B,当C为'Q'的时候, 表示这是一条询问操作，他询问ID从A到B（包括A,B）的学生当中，成绩最高的是多少

# 当C为‘U’的时候，表示这是一条更新操作，要求把ID为A的学生的成绩更改为B。

m,n = map(int,input().split())

glist = list(map(int, input().split()))

x = []

for i in range(n):

op\_list = list(input().split())

if op\_list[0] == 'Q':

num1, num2 = int(op\_list[1]), int(op\_list[2])

num1, num2= min(num1, num2), max(num1, num2)

glist1 = glist[num1-1:num2]

# print(glist1)

x.append(max(glist1))

if op\_list[0] == 'U':

num1, num2 = int(op\_list[1]), int(op\_list[2])

glist[num1-1] = num2

# print(glist)

for i in range(len(x)):

print(x[i])

**huawei2016\_2.py**

# coding = gbk

# import sys

# for line in sys.stdin:

# a = line.split()

# print(int(a[0]) + int(a[1]))

#

# output\_list = []

# filename\_dict = {}

# while True:

# s = input()

# if s == '':

# break

# else:

# s = list(s.split())

# file\_name = list(s[0].split('\\'))[-1]

# filename = file\_name + ' ' + s[1]

# if filename not in filename\_dict:

# filename\_dict[filename] = 1

# else:

# num = filename\_dict[filename] + 1

# filename\_dict[filename] = num

# if len(file\_name) > 16:

# file\_name = file\_name[-16:]

# output\_list.remove(file\_name + ' ' + s[1] + " " + str(filename\_dict[filename]-1))

# if len(file\_name) > 16:

# file\_name = file\_name[-16:]

# output = file\_name + ' ' + s[1] + " " + str(filename\_dict[filename])

# output\_list.append(output)

#

# for i in range(len(output\_list)):

# print(output\_list[i])

output\_list = []

filename\_dict = {}

import sys

for line in sys.stdin:

if ord(line[0]) == 10:

break

else:

s = list(line.split())

file\_name = list(s[0].split('\\'))[-1]

filename = file\_name + ' ' + s[1]

if filename not in filename\_dict:

filename\_dict[filename] = 1

else:

num = filename\_dict[filename] + 1

filename\_dict[filename] = num

if len(file\_name) > 16:

file\_name = file\_name[-16:]

output\_list.remove(file\_name + ' ' + s[1] + " " + str(filename\_dict[filename]-1))

if len(file\_name) > 16:

file\_name = file\_name[-16:]

output = file\_name + ' ' + s[1] + " " + str(filename\_dict[filename])

output\_list.append(output)

for i in range(len(output\_list)):

print(output\_list[i])

**huawei2016\_3.py**

# coding=gbk

# 扑克牌游戏大家应该都比较熟悉了，一副牌由54张组成，含3~A，2各4张，小王1张，大王1张。牌面从小到大用如下字符和字符串表示（其中，小写joker表示小王，大写JOKER表示大王）:)

# 3 4 5 6 7 8 9 10 J Q K A 2 joker JOKER

# 输入两手牌，两手牌之间用“-”连接，每手牌的每张牌以空格分隔，“-”两边没有空格，如：4 4 4 4-joker JOKER

# 请比较两手牌大小，输出较大的牌，如果不存在比较关系则输出ERROR

#

# 基本规则：

# （1）输入每手牌可能是个子，对子，顺子（连续5张），三个，炸弹（四个）和对王中的一种，不存在其他情况，由输入保证两手牌都是合法的，顺子已经从小到大排列；

# （2）除了炸弹和对王可以和所有牌比较之外，其他类型的牌只能跟相同类型的存在比较关系（如，对子跟对子比较，三个跟三个比较），不考虑拆牌情况（如：将对子拆分成个子）

# （3）大小规则跟大家平时了解的常见规则相同，个子，对子，三个比较牌面大小；顺子比较最小牌大小；炸弹大于前面所有的牌，炸弹之间比较牌面大小；对王是最大的牌；

# （4）输入的两手牌不会出现相等的情况。

#

# 答案提示：

# （1）除了炸弹和对王之外，其他必须同类型比较。

# （2）输入已经保证合法性，不用检查输入是否是合法的牌。

# （3）输入的顺子已经经过从小到大排序，因此不用再排序了.

# import sys

# for line in sys.stdin:

# a = line.split()

# print(int(a[0]) + int(a[1]))

m, n = input().split('-')

x = list(m.split())

y = list(n.split())

poker\_dict = {'3':3,'4':4, '5':5, '6':6, '7':7 ,'8':8 , '9':9, '10':10,

'J':11, 'Q':12, 'K':13, 'A':14, '2':15, 'joker':'joker', 'JOKER':'JOKER'}

def judge\_xy(x,y):

w = x

z = y

for i in range(len(x)):

w[i] = poker\_dict[x[i]]

for j in range(len(y)):

z[j] = poker\_dict[y[j]]

if w == ['joker', 'JOKER'] or z == ['joker', 'JOKER']:

return x if w == ['joker', 'JOKER'] else y

if len(w) != len(z):

if len(w) == 4 or len(z) == 4:

return x if len(w) == 4 else y

else:

return 'ERROR'

if len(w) == len(z):

return x if w[-1] > z[-1] else y

s = judge\_xy(x, y)

b = ' '

if s == 'ERROR':

print('ERROR')

else:

# output = b.join(s)

output = m if s == x else n

print(output)

**huawei2018\_ 1.py**

# coding = gbk

# 括号匹配

# 给定一个字符串，里边可能包含“()”、“[]”、“{}”三种括号，请编写程序检查该字符串中的括号是否成对出现，且嵌套关系正确。

# 输出：true:若括号成对出现且嵌套关系正确，或该字符串中无括号字符；

# false:若未正确使用括号字符。

# 实现时，无需考虑非法输入。

# 输入描述：

# 输入为：

# 字符串

# 例子：(1+2)/(0.5+1)

# 输出描述：

# 输出为：true

#

# 思路：栈

# 遇到左符号，则压入，遇到右符号，弹出顶层的符号和右符号比对，如果符合，则继续，

# 否则输出false

s = input()

# s1 = list(s)

# voc1 = ['(', '[', '{']

# voc2 = [')', ']', '}']

# def judge\_s(s):

# count1 = 0

# count2 = 0

# count3 = 0

# for i in s:

# if (i == ')' and count1 == 0) or (i == ']' and count2 == 0) or (i == '}' and count3 == 0):

# return False

# elif i == '(':

# count1 += 1

# elif i == ')':

# count1 -= 1

# elif i == '[':

# count2 += 1

# elif i == ']':

# count2 -= 1

# elif i == '{':

# count3 += 1

# elif i == '}':

# count3 -= 1

#

# if count1 == 0 and count2 == 0 and count3 == 0:

# return True

# else:

# return False

def judge\_rl(a, b):

if a == '(' and b == ')':

return 1

if a == '[' and b == ']':

return 1

if a == '{' and b == '}':

return 1

else:

return 0

def judge\_s(s):

x = []

for i in s:

if i == '(' or i == '[' or i == '{':

x.append(i)

elif i == ')' or i == ']' or i == '}':

if x != [] and judge\_rl(x[-1], i) == 1:

return True

else:

return False

break

print(judge\_s(s))

**huawei2018\_2.py**

# coding = gbk

# 平安果

# 简要描述：

# 给定一个M行N列的矩阵（M\*N个格子），每个格子中放着一定数量的平安果。

# 你从左上角的各自开始，只能向下或者向右走，目的地是右下角的格子。

# 每走过一个格子，就把格子上的平安果都收集起来。求你最多能收集到多少平安果。

# 注意：当经过一个格子时，需要一次性把格子里的平安果都拿走。

# 限制条件：1<N,M<=50；每个格子里的平安果数量是0到1000（包含0和1000）.

# 输入描述：

# 输入包含两部分：

# 第一行M, N

# 接下来M行，包含N个平安果数量

# 输出描述：

# 一个整数

# 最多拿走的平安果的数量

# 示例：

# 输入

# 2 4

# 1 2 3 40

# 6 7 8 90

# 输出

# 136

#

# 思路：动态规划

# 动态方程：当前位置能够获得的最大苹果数=max(从上面走能够获得最大苹果+从左边走能获得最大苹果）

# dp(0,0)=app[0][0]

import numpy as np

m,n = map(int, input().split())

x = np.zeros((m,n))

for i in range(m):

x[i] = list(map(int,input().split()))

def max\_x(m,n,x):

if m == n == 0:

return x[0][0]

if m == 0:

return x[m][n] + max\_x(m, n-1, x)

# print(res)

if n == 0:

return x[m][n] + max\_x(m-1, n, x)

# print(res)

else:

return max(x[m][n] + max\_x(m-1,n,x), x[m][n] + max\_x(m, n-1, x))

def max\_x(m,n,x):

if m == 0 and n == 0:

return x[0][0]

elif m == 0:

return x[m][n] + max\_x(m, n-1, x)

elif n == 0:

return x[m][n] + max\_x(m-1, n ,x)

else:

return max(x[m][n] + max\_x(m, n-1, x), x[m][n] + max\_x(m-1, n, x))

res = max\_x(m-1,n-1,x)

print(res)

**huawei2019\_1.py**

# coding = gbk

s = input()

s = list(s)

add = []

minus = []

for i in range(len(s)):

if s[i] == "+":

add.append(i)

s[i] = " "

elif s[i] == "-":

minus.append(i)

s[i] = " "

s = ''.join(s)

alist = list(map(int, s.split()))

a\_m = add+minus

a\_m = sorted(a\_m)

res = alist[0]

for j in range(len(a\_m)):

if a\_m[j] in add:

res += alist[j+1]

else:

res -= alist[j+1]

print(res)

**huawei2019\_2.py**

# coding = gbk

import numpy as np

s = input()

voc = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z',

'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']

num = np.zeros(26)

# s1 = ''

# for i in s:

# if i not in s1:

# s1 += i

s1 = set(s)

slist = list(s1)

for i in range(0, 26):

if voc[i] in slist and voc[i+26] in slist:

num[i] = 1

# print(num)

# def find\_voc(s):

sum\_num = sum(num)

res\_list = []

while sum\_num > 0:

res = ''

max\_len = 1

zero\_list = [-1]

sum\_num = sum(num)

for i in range(26):

if num[i] == 0:

zero\_list.append(i)

# print(zero\_list)

for j in range(len(zero\_list)-1):

length = zero\_list[j+1] - zero\_list[j]

if max\_len < length:

max\_len = length

for l in range(1, max\_len):

res += (voc[zero\_list[j] + l] + voc[zero\_list[j] + l + 26])

res\_list.append(res)

# print(res\_list)

res = list(res)

for i in range(len(res)):

if res[i] in slist:

slist.remove(res[i])

# print(slist)

num = np.zeros(26)

for i in range(0, 26):

if voc[i] in slist and voc[i+26] in slist:

num[i] = 1

# print(num)

sum\_num = sum(num)

# print(slist)

for i in range(len(res\_list)):

print(res\_list[i])

# print(max\_len)

# print(res)

**Huawei2019\_3.py**

import sys

n = int(sys.stdin.readline().strip())

a = [n]

for i in range(n):

value = list(map(int, sys.stdin.readline().split()))

a.append(value)

# print(a)

def fun(a):

if a[0] == 1:

return 1

a\_temp = a[2:]

ha = list()

ha.append(a[1])

cut = list()

cut.append([0,0,0,0])

flag = False

for a, b in a\_temp:

for i,(c,d) in enumerate(ha):

if a == c:

flag = True

if cut[i][0] == 0:

cut[i][0] = 1

break

if b == d:

flag = True

if cut[i][1] == 0:

cut[i][1] = 1

break

if (c-a) == (b-d):

flag = True

if cut[i][2] == 0:

cut[i][2] = 1

break

if (a-c) == (b-d):

flag = True

if cut[i][3] == 0:

cut[i][3] = 1

break

if not flag:

ha.append([a, b])

cut.append([0,0,0,0])

flag = False

num = 0

for h,s,p,n in cut:

if [h,s,p,n] == [0,0,0,0]:

num += 1

else:

num += (h+s+p+n)

return num

n = fun(a)

print(n)

**tengxun\_test1.py**

def deleteZeroOne(s):

i = 0

while i < len(s)-1 and len(s) >= 2:

if s[i] == '1' and s[i+1] == '0':

s = s[:i] + s[i+2:]

else:

i += 1

return s

def deleteAgain(s):

while( s != deleteZeroOne(s)):

s = deleteZeroOne(s)

return s

s = '1101010001'

print(deleteAgain(s))

**zuiyou\_test1.py**

def match(s,source):

if len(source) == 0:

return False

if len(source) == 1:

if s in source[0]:

return True

else:

return False

s\_start = []

for i in range(len(source)-1):

s\_start.append(source[i])

s\_start.append(source[i][0])

s\_end = []

for i in range(len(source[-1])):

se = source[-1][:i+1]

s\_end.append(se)

while s != '':

if s\_start != [] and s[0:len(source[0])] == s\_start[0]:

s = s[len(source[0]):]

s\_start.remove(s\_start[0])

s\_start.remove(s\_start[0])

source.remove(source[0])

elif s\_start != [] and s[0] == s\_start[1]:

s = s[1:]

s\_start.remove(s\_start[0])

s\_start.remove(s\_start[0])

source.remove(source[0])

elif s\_start == [] and s in s\_end:

return True

else:

return False

return True

if \_\_name\_\_ == "\_\_main\_\_":

source = ["wang", "hai", "bao"]

s = "whb" # "wanghb" "wanghbao" "wanghaiba" 'wh'

print(match(s,source))

**zuiyou\_test.py**

source = ['wang', 'hai', 'bao']

string = "whb"

def match(source, string):

# print(source, string)

if len(source) == 0:

return False

if len(source) == 1:

if string in source[0]:

return True

else:

return False

judge1, judge2 = False, False

if source[0] == string[:len(source[0])]:

judge1 = match(source[1:], string[len(source[0]):])

if source[0][0] == string[0]:

judge2 = match(source[1:], string[1:])

return judge1 or judge2

print(match(source, string))