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))
```