Олимпиада СПбГУ по информатике 2022/23 учебного года

A	B	C	D	E	F	Sum
100	100	100	10	52	10	372

Task A ()

```
d = list(map(lambda x: int(x) - 1, input().split()))
a = []
for (x,i) in enumerate(d):
    a = a[:i] + [x] + a[i:]
b = [-1] * 6
for (i,x) in enumerate(a):
    b[x] = str(i + 1)
print(' '.join(b))
```

Task B ()

```
mod = 20011
first = input() == 'first'
n = int(input())
a = sum(map(int,input().split()))
if first:
    for i in range(mod):
        if (i * n) % mod == a:
            print(i)
else:
    print(a % mod)
```

Task C ()

```
from copy import deepcopy
answer = set()
def dfs(a,visited,i,j):
    n = len(a)
    m = len(a[0])
    #print(i,j,n,m)
    if visited[i][j]:
        return set()
    visited[i][j] = True
    cells = {(i,j)}
    if i > 0 and a[i - 1][j] == a[i][j]:
        cells |= dfs(a,visited,i - 1,j)
    if i < n - 1 and a[i + 1][j] == a[i][j]:
        cells |= dfs(a,visited,i + 1,j)
    if j > 0 and a[i][j - 1] == a[i][j]:
        cells |= dfs(a,visited,i,j - 1)
    if j < m - 1 and a[i][j + 1] == a[i][j]:
        cells |= dfs(a,visited,i,j + 1)
    return cells
def count_rects(a):
    '''
␣␣␣␣for␣i␣in␣a:
␣␣␣␣␣␣␣␣for␣j␣in␣i:
␣␣␣␣␣␣␣␣␣␣␣␣print(j,end␣=␣'')
␣␣␣␣␣␣␣␣print()
␣␣␣␣print()
␣␣␣␣'''
    global answer
    n = len(a)
    m = len(a[0])
    visited = [[False] * m for i in range(n)]
    for i in range(n):
        for j in range(m):
            if visited[i][j]:
                continue
            #print(i,j)
            cells = dfs(a,visited,i,j)
            #print(cells)
            x1 = min(map(lambda x: x[0],cells))
            x2 = max(map(lambda x: x[0],cells)) + 1
            y1 = min(map(lambda x: x[1],cells))
            y2 = max(map(lambda x: x[1],cells)) + 1
            if len(cells) == (y2 - y1) * (x2 - x1):
                #print(x1,x2,y1,y2)
                h = min(x2 - x1,y2 - y1)
                w = max(x2 - x1,y2 - y1)
                answer.add((h,w))
def test3(a,x3,y3):
    n = len(a)
    m = len(a[0])
    for i in range(n - x3 + 1):
        for j in range(m - y3 + 1):
            #print(i,j)
            t = set()
            for i1 in range(i, i + x3):
                t = t | set(a[i1][j:j+y3])
            if len(t) > 1:
                continue
            x = t.pop()
            #print(j - 1, j + y3)
            if i != 0 and i != n - x3:
                t = set()
                for i1 in (i - 1, i + x3):
                    for j1 in range(j, j + y3):
                        t.add(a[i1][j1])
                #print(i,j,t)
                if len(t) == 1:
                    continue
            if j != 0 and j != m - y3:
                t = set()
                for j1 in (j - 1, j + y3):
                    for i1 in range(i, i + x3):
```

```
                        t.add(a[i1][j1])
                #print(i,j,t)
                if len(t) == 1:
                    continue
            a1 = deepcopy(a)
            for i1 in range(i, i + x3):
                for j1 in range(j, j + y3):
                    a1[i1][j1] = 3
            count_rects(a1)
def test(x1,y1,x2,y2,x3,y3):
    # (0,0), (x1,y1)
    a = [[1] * y1 for _ in range(x1)]
    count_rects(a)
    if x2 <= x1 and y2 <= y1:
        for i in range(x2):
            for j in range(y2):
                a[i][j] = 2
        count_rects(a)
        test3(a,x3,y3)
        test3(a,y3,x3)
        a = [[1] * y1 for _ in range(x1)]
        for i in range(x1 - x2,x1):
            for j in range(y2):
                a[i][j] = 2
        count_rects(a)
        test3(a,x3,y3)
        test3(a,y3,x3)
        a = [[1] * y1 for _ in range(x1)]
        for i in range(x2):
            for j in range(y1 - y2,y1):
                a[i][j] = 2
        count_rects(a)
        test3(a,x3,y3)
        test3(a,y3,x3)
        a = [[1] * y1 for _ in range(x1)]
        for i in range(x1 - x2,x1):
            for j in range(y1 - y2,y1):
                a[i][j] = 2
        count_rects(a)
        test3(a,x3,y3)
        test3(a,y3,x3)
        a = [[1] * y1 for _ in range(x1)]
    if y2 <= x1 and x2 <= y1:
        for i in range(y2):
            for j in range(x2):
                a[i][j] = 2
        count_rects(a)
        test3(a,x3,y3)
        test3(a,y3,x3)
        a = [[1] * y1 for _ in range(x1)]
        for i in range(x1 - y2,x1):
            for j in range(x2):
                a[i][j] = 2
        count_rects(a)
        test3(a,x3,y3)
        test3(a,y3,x3)
        a = [[1] * y1 for _ in range(x1)]
        for i in range(y2):
            for j in range(y1 - x2,y1):
                a[i][j] = 2
        count_rects(a)
        test3(a,x3,y3)
        test3(a,y3,x3)
        a = [[1] * y1 for _ in range(x1)]
        for i in range(x1 - y2,x1):
            for j in range(y1 - x2,y1):
                a[i][j] = 2
        count_rects(a)
        test3(a,x3,y3)
        test3(a,y3,x3)
        a = [[1] * y1 for _ in range(x1)]
a1,b1 = map(int,input().split())
a2,b2 = map(int,input().split())
a3,b3 = map(int,input().split())
```

```
test(a1,b1,a2,b2,a3,b3)
test(a1,b1,a3,b3,a2,b2)
test(a2,b2,a1,b1,a3,b3)
test(a2,b2,a3,b3,a1,b1)
test(a3,b3,a1,b1,a2,b2)
test(a3,b3,a2,b2,a1,b1)
for x,y in sorted(answer):
    print(x,y)
```

Task D ()

```
n = int(input())
a = list(map(int,input().split()))
d = {}
def set(a1,restart,x):
    global d
    d[(tuple(a1),tuple(restart))] = x
def get(a1,restart):
    return d[(tuple(a1),tuple(restart))]
def has(a1,restart):
    return (tuple(a1),tuple(restart)) in d
def process(a1,restart):
    if has(a1, restart):
        return
    a_new = a1[:]
    res_new = restart[:]
    for i in range(n):
        for j in range(a1[i]):
            a_new[i] = j
            process(a_new, res_new)
            if get(a_new, res_new) == (-2,-1):
                set(a1,restart,(i, a1[i] - j))
                return
            a_new[i] = a1[i]
    for i in range(n):
        if restart[i]:
            a_new[i] = a[i]
            res_new[i] = False
            process(a_new, res_new)
            if get(a_new, res_new) == (-2,-1):
                set(a1,restart,(i, 0))
                return
            a_new[i] = a1[i]
            res_new[i] = True
    set(a1,restart,(-2,-1))
a1 = a[:]
restart = [True] * n
while True:
    #print(a1,restart)
    process(a1, restart)
    i,x = get(a1, restart)
    print(i + 1,x)
    if x == -1:
        break
    if x == 0:
        a1[i] = a[i]
        restart[i] = False
    else:
        a1[i] -= x
    i,x = map(int,input().split())
    i -= 1
    if x == -1:
        break
    if x == 0:
        a1[i] = a[i]
        restart[i] = False
    else:
        a1[i] -= x
```

Task E ()

```
count = 0
d1 = {}
d2 = {}
m = 10
def process(n,a):
    global count,d1,d2
    if n == m:
        #print(a)
        count += 1
        a = tuple(a[1:])
        d1[count] = a
        d2[a] = count
        return
    for i in range(a[-1], m + 1):
        process(n + 1, a + [i])
process(0,[0])
t = int(input())
if input() == 'transmit':
    for _ in range(t):
        n = int(input())
        x = d1[n]
        for i in x:
            print('1' * i + '0' * (m - i))
else:
    for _ in range(t):
        input()
        a = [input() for _ in range(m)]
        counts = list(map(lambda x: x.count('1'), a))
        x = tuple(sorted(counts))
        #print(a,counts,x)
        print(d2[x])
```

Task F ()

```
a = input()
b = input()
def intervals(a):
    n = len(a)
    l = 0
    i = n - 1
    inter = []
    while i >= 0:
        if a[i] == ')':
            i_last = i
            while a[i_last] != '(':
                i_last -= 1
            x,il = map(int,a[i_last + 1:i].split('|'))
            inter.append((l, l + il, x))
            l += il
            i = i_last - 1
        else:
            inter.append((l, l + 1, int(a[i])))
            l += 1
            i -= 1
    inter.append((l, 10 ** 18, 0))
    return inter
inter_a = intervals(a)
inter_b = intervals(b)
inter_total = []
i = 0
j = 0
while i < len(inter_a) and j < len(inter_b):
    if inter_a[i][1] == inter_b[j][1]:
        inter_total.append((inter_a[i][1], inter_a[i][2] + inter_b[j][2]))
        i += 1
        j += 1
    elif inter_a[i][1] < inter_b[j][1]:
        inter_total.append((inter_a[i][1], inter_a[i][2] + inter_b[j][2]))
        i += 1
    else:
        inter_total.append((inter_b[j][1], inter_a[i][2] + inter_b[j][2]))
        j += 1
s = ''
l_last = 0
carry_last = False
for l,x in inter_total[:-1]:
    leg = l - l_last
    carry = x >= 10 or (x == 9 and carry_last)
    s = str((x + carry_last) % 10) + s
    if leg < 6:
        s = str((x + carry) % 10) * (leg - 1) + s
    else:
        s = '(' + str((x + carry) % 10) + '|' + str(leg - 1) + ')' + s
    l_last = l
    carry_last = carry
if carry_last:
    s = '1' + s
print(s)
```