Олимпиада «Ломоносов» по информатике
2025-2026 учебный год. Заключительный этап
Работа участника с id заявки 1601885, логином inf26f_312
Сводный итог по всем задачам в проверяющей системе

RunID	Time	Username	Prob	Lang	Result	Tests	Score
193	2:54:08	inf26f_312	2	pypy3	OK	53	100
168	2:37:56	inf26f_312	6	pypy3	OK	21	100
140	2:13:42	inf26f_312	5	pypy3	OK	103	100
134	2:10:41	inf26f_312	4	pypy3	OK	103	100
128	2:07:27	inf26f_312	3	pypy3	OK	23	100
109	1:53:39	inf26f_312	1	pypy3	OK	23	100

600 (шестьсот) технических баллов
100 (сто) итоговых баллов

председатель Жюри кандидат физ.-мат. наук Малышко В. В.

зам. председателя Жюри кандидат физ.-мат. наук Корныхин Е. В.

Посылка по задаче 1

```
[1] import sys
[2]
[3]
[4] def a():
[5]     vvod = sys.stdin.read().split()
[6]     if not vvod:
[7]         return
[8]     K = int(vvod[0])
[9]     strings = vvod[1:K + 1]
[10]    m = -1
[11]    d = float('inf')
[12]    for s in strings:
[13]        parts = s.split('.')
[14]        tot = 0
[15]        for part in parts:
[16]            digit_val = 0
[17]            m_digit = 0
[18]            for char in reversed(part):
[19]                if char == 'n':
[20]                    val = 0
[21]                elif char == 'i':
[22]                    val = 1
[23]                elif char == 'v':
[24]                    val = 5
[25]                elif char == 'x':
[26]                    val = 10
[27]                else:
[28]                    val = 0
[29]                if val < m_digit:
[30]                    digit_val -= val
[31]                else:
[32]                    digit_val += val
[33]                    m_digit = val
[34]            tot = tot * 26 + digit_val
[35]        if tot > m:
[36]            m = tot
[37]        if tot < d:
[38]            d = tot
[39]    def perevod(n):
[40]        if n == 0:
[41]            return 'a'
[42]        res = []
[43]        while n > 0:
[44]            res.append(chr(ord('a') + n % 26))
[45]            n //= 26
[46]        return ''.join(reversed(res))
[47]
[48]    print(perevod(m))
[49]    print(perevod(d))
[50]
[51]
[52]if __name__ == '__main__':
[53]    a()
```

Посылка по задаче 2

```
[1]  import sys
[2]  from math import isqrt
[3]  def proverka_prostoe(n):
[4]      if n < 2:
[5]          return False
[6]      if n < 4:
[7]          return True
[8]      if n % 2 == 0 or n % 3 == 0:
[9]          return False
[10]     i = 5
[11]     while i * i <= n:
[12]         if n % i == 0 or n % (i + 2) == 0:
[13]             return False
[14]         i += 6
[15]     return True
[16] def proverka_brazilskoe(n):
[17]     for b in range(2, n - 1):
[18]         k = 0
[19]         t = n
[20]         vse_edinicy = True
[21]         while t > 0:
[22]             if t % b != 1:
[23]                 vse_edinicy = False
[24]                 break
[25]             t //= b
[26]             k += 1
[27]         if vse_edinicy and k >= 3:
[28]             return True
[29]     return False
[30] def proverka_brazilskoe_bystro(n):
[31]     for b in range(2, n - 1):
[32]         s = 1 + b + b * b
[33]         if s > n:
[34]             break
[35]         stepen = b * b * b
[36]         while s < n:
[37]             s += stepen
[38]             stepen *= b
[39]         if s == n:
[40]             return True
[41]     return False
[42] data = sys.stdin.read().split()
[43] n = int(data[0])
[44] chisla = list(map(int, data[1:1+n]))
[45] mnozhestvo = set()
[46] for x in chisla:
[47]     if proverka_prostoe(x) and proverka_brazilskoe_bystro(x):
[48]         mnozhestvo.add(x)
[49]
[50] print(len(mnozhestvo))
[51]
```

Посылка по задаче 3

```
[1]  n = int(input())
[2]  c = int(input())
[3]  r = int(input())
[4]  row = r - 1
[5]  col = c - 1
[6]  k = min(row, col, n - 1 - row, n - 1 - col)
[7]  offset = 0
[8]  for i in range(k):
[9]      offset += 4 * (n - 1 - 2 * i)
[10] side = n - 2 * k
[11] if side == 1:
[12]     print(offset)
[13] else:
[14]     lr = row - k
[15]     lc = col - k
[16]     if lr == 0:
[17]         pos = lc
[18]     elif lc == side - 1:
[19]         pos = (side - 1) + lr
[20]     elif lr == side - 1:
[21]         pos = (side - 1) + (side - 1) + (side - 1 - lc)
[22]     else:
[23]         pos = 3 * (side - 1) + (side - 1 - lr)
[24]     print(offset + pos)
```

Посылка по задаче 4

```
[1]  import bisect
[2]  def funcia():
[3]      n = int(input())
[4]      a = list(map(int, input().split()))
[5]      t = []
[6]      for x in a:
[7]          pos = bisect.bisect_right(t, x)
[8]          if pos == len(t):
[9]              t.append(x)
[10]         else:
[11]             t[pos] = x
[12]     print(n - len(t))
[13] funcia()
```

Посылка по задаче 5

```
[1] import sys
[2]
[3]
[4] def funcia():
[5]     vvod = sys.stdin.buffer.read().split()
[6]     idx = 0
[7]     n = int(vvod[idx]);
[8]     idx += 1
[9]     mountains = []
[10]    for i in range(n):
[11]        x = int(vvod[idx]);
[12]        idx += 1
[13]        y = int(vvod[idx]);
[14]        idx += 1
[15]        h = int(vvod[idx]);
[16]        idx += 1
[17]        mountains.append((x, y, h))
[18]    x0, y0, h0 = mountains[0]
[19]    candidates = []
[20]    for i in range(1, n):
[21]        x, y, h = mountains[i]
[22]        if x > x0 and y > y0 and h < h0:
[23]            candidates.append((x, y, h))
[24]    if not candidates:
[25]        print(1)
[26]        return
[27]    candidates.sort(key=lambda m: (m[0], m[1], -m[2]))
[28]    m = len(candidates)
[29]    dp = [1] * m
[30]    for i in range(m):
[31]        for j in range(i):
[32]            if candidates[j][0] < candidates[i][0] and \
[33]                    candidates[j][1] < candidates[i][1] and \
[34]                    candidates[j][2] > candidates[i][2]:
[35]                if dp[j] + 1 > dp[i]:
[36]                    dp[i] = dp[j] + 1
[37]    print(max(dp) + 1)
[38] funcia()
```

Посылка по задаче 6

```
[1]  import sys
[2]  from collections import deque
[3]  from math import factorial
[4]  def prochitat_vvod():
[5]      lines = sys.stdin.buffer.read().decode().splitlines()
[6]      if not lines:
[7]          return None
[8]      toks = []
[9]      idx = 0
[10]     while idx < len(lines) and len(toks) < 6:
[11]         toks += lines[idx].split()
[12]         idx += 1
[13]     if len(toks) < 6:
[14]         return None
[15]     k1, k2, k3, k4, n, m = map(int, toks[:6])
[16]     pole = []
[17]     while idx < len(lines) and len(pole) < n:
[18]         s = lines[idx].rstrip("\n")
[19]         if len(s) == 0 and m == 0:
[20]             s = ""
[21]         pole.append(s)
[22]         idx += 1
[23]     return k1, k2, k3, k4, n, m, pole
[24] def nayti_komponenty(zanyatye):
[25]     ost = set(zanyatye)
[26]     comps = []
[27]     while ost:
[28]         start = next(iter(ost))
[29]         q = deque([start])
[30]         ost.remove(start)
[31]         comp = [start]
[32]         while q:
[33]             r, c = q.popleft()
[34]             for dr, dc in ((1, 0), (-1, 0), (0, 1), (0, -1)):
[35]                 nr, nc = r + dr, c + dc
[36]                 p = (nr, nc)
[37]                 if p in ost:
[38]                     ost.remove(p)
[39]                     q.append(p)
[40]                     comp.append(p)
[41]         comps.append(comp)
[42]     return comps
[43] def reshit_komponentu(cells, limity):
[44]     lim1, lim2, lim3, lim4 = limity
[45]     k = len(cells)
[46]     idx_of = {p: i for i, p in enumerate(cells)}
[47]     cellset = set(cells)
[48]     spiski = [[] for _ in range(k)]
[49]     dobavleno = set()
[50]     for (r, c) in cells:
[51]         for L in (1, 2, 3, 4):
[52]             for dr, dc in ((0, 1), (1, 0)):
[53]                 if L == 1 and dr == 1:
[54]                     continue
[55]                 ok = True
[56]                 mask = 0
[57]                 coords = []
[58]                 for t in range(L):
[59]                     p = (r + dr * t, c + dc * t)
[60]                     if p not in cellset:
[61]                         ok = False
[62]                         break
[63]                     coords.append(p)
[64]                 if not ok:
[65]                     continue
[66]                 for p in coords:
[67]                     mask |= 1 << idx_of[p]
[68]                 key = (mask, L)
[69]                 if key in dobavleno:
[70]                     continue
[71]                 dobavleno.add(key)
[72]                 for p in coords:
[73]                     spiski[idx_of[p]].append((mask, L))
[74]     memo = {}
[75]     def rek(mask):
```

```
        if mask == 0:
            return {(0, 0, 0, 0): 1}
        if mask in memo:
            return memo[mask]
        i = (mask & -mask).bit_length() - 1
        res = {}
        for segmask, L in spiski[i]:
            if segmask & mask != segmask:
                continue
            pod = rek(mask ^ segmask)
            for (a1, a2, a3, a4), val in pod.items():
                if L == 1:
                    na1, na2, na3, na4 = a1 + 1, a2, a3, a4
                elif L == 2:
                    na1, na2, na3, na4 = a1, a2 + 1, a3, a4
                elif L == 3:
                    na1, na2, na3, na4 = a1, a2, a3 + 1, a4
                else:
                    na1, na2, na3, na4 = a1, a2, a3, a4 + 1
                if na1 > lim1 or na2 > lim2 or na3 > lim3 or na4 > lim4:
                    continue
                key = (na1, na2, na3, na4)
                res[key] = res.get(key, 0) + val
        memo[mask] = res
        return res
    polinom = rek((1 << k) - 1)
    otf = {}
    for u, v in polinom.items():
        if u[0] <= lim1 and u[1] <= lim2 and u[2] <= lim3 and u[3] <= lim4:
            otf[u] = v
    return otf
def main():
    inp = prochitat_vvod()
    if inp is None:
        return
    k1, k2, k3, k4, n, m, pole = inp
    zanyatye = []
    for i in range(n):
        row = pole[i]
        for j, ch in enumerate(row):
            if ch == '#':
                zanyatye.append((i, j))

    K = len(zanyatye)
    if K != k1 + 2 * k2 + 3 * k3 + 4 * k4:
        print(0)
        return
    if K == 0:
        print(1 if (k1, k2, k3, k4) == (0, 0, 0, 0) else 0)
        return
    komponenty = nayti_komponenty(zanyatye)
    dp = {(0, 0, 0, 0): 1}
    limity = (k1, k2, k3, k4)
    for comp in komponenty:
        pol = reshit_komponentu(comp, limity)
        nov = {}
        for (a1, a2, a3, a4), v1 in dp.items():
            for (b1, b2, b3, b4), v2 in pol.items():
                na1, na2, na3, na4 = a1 + b1, a2 + b2, a3 + b3, a4 + b4
                if na1 <= k1 and na2 <= k2 and na3 <= k3 and na4 <= k4:
                    key = (na1, na2, na3, na4)
                    nov[key] = nov.get(key, 0) + v1 * v2
        dp = nov
        if not dp:
            print(0)
            return
    bez_metok = dp.get((k1, k2, k3, k4), 0)
    ans = bez_metok * factorial(k1) * factorial(k2) * factorial(k3) * factorial(k4)
    print(ans)



main()
```