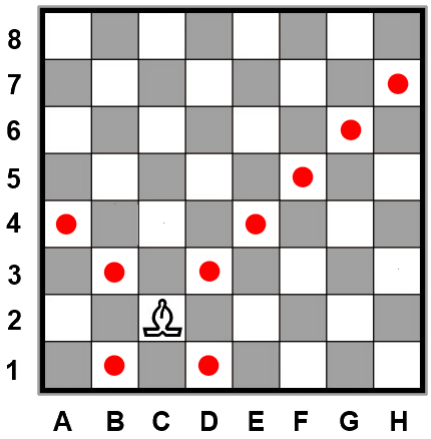
Задача 1. Ход слоном

Входной файл	стандартный ввод / input.txt
Выходной файл	стандартный вывод / output.txt
Ограничение времени (сек/тест)	1
Ограничение памяти (Мб)	64

Условие задачи

Чтобы защитить свой блог от автоматического спама, Вовочка решил использовать оригинальную капчу (CAPTCHA – англ. Completely Automated Public Turing test to tell Computers and Humans Apart – полностью автоматизированный публичный тест Тьюринга для различения компьютеров и людей). Как большой любитель шахмат Вовочка в качестве теста придумал просить людей решить простую задачу, а именно ответить на вопрос, сколько различных ходов можно сделать шахматной фигурой, находящейся на пустой доске в определенной позиции. Для проверки ответа требуется составить программу, сегодня Вовочка реализует алгоритм проверки ответа для Слона и просит ему помочь.



Как известно, Слон – шахматная фигура, которая, не имея препятствий, может перемещаться на любую клетку, находящуюся на любой из двух диагоналей. Позиция фигуры (клетка) в шахматах задается в виде пары – латинской буквы от A до H и цифры от 1 до 8. На рисунке справа показан пример доски, поясняющий правила обозначения клеток. На данном рисунке единственная фигура Слон находится в клетке C2. Точками отвечены клетки, на которые Слон может пойти с позиции C2.

Входные данные

В единственной строке входного файла в шахматной нотации (буква латинского алфавита от A до H и цифра от 1 до 8) указана клетка, на которой находится Слон. Буквы латинского алфавита – заглавные.

Выходные данные

В выходной файл необходимо записать единственное число – количество различных клеток, на которые может за один ход переместиться Слон, находящийся в заданной позиции на пустой шахматной доске.

Пример входного файла (stdin / input.txt)	Пример выходного файла (stdout / output.txt)	Комментарий
C2	9	Пример соответствует рисунку в условии задачи.
D6	11	

Решение жюри

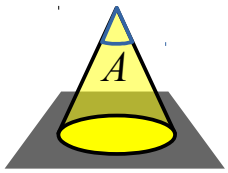
```
#define _USE_MATH_DEFINES

#include <fstream>
#include <string>
#include <cmath>

using namespace std;


int main()
{
  ifstream cin("input.txt");
  ofstream cout("output.txt");

  string letters = "ABCDEFGH",
         digits = "12345678";

  string pos;
  cin >> pos;

  int r0 = letters.find(pos[0]),
      c0 = digits.find(pos[1]);

  int n = 0;
  for (int r = 0; r < 8; r++)
    for (int c = 0; c < 8; c++)
      if (!(r == r0 && c == c0) &&
          abs(r - r0) == abs(c - c0))
        n++;

  cout << n << endl;
}
```

Задача 2. Освещение коридора

Входной файл	стандартный ввод / input.txt
Выходной файл	стандартный вывод / output.txt
Ограничение времени (сек/тест)	1
Ограничение памяти (Мб)	64

Условие задачи

В большой новой квартире, в которую недавно переехал Вовочка, имеется длинный прямой коридор, соединяющий все помещений квартиры. Вовочка решил сделать в квартире небольшой ремонт и, в частности, натянуть натяжной потолок и переделать освещение в коридоре.



Для организации освещения Вовочка собирается использовать встраиваемые потолочные энергосберегающие светильники, обеспечивающие направленное освещение с углом рассеивания ***A***. Для простоты будем считать, что это означает, что свет равномерно расходится от светильника в виде конуса с углом раствора (углом между образующими конуса) A. Опять же для простоты примем, что за пределы угла А свет не выходит.

Для красоты Вовочка хочет все светильники разместить по центру коридора. Другими словами все светильники должны располагаться на воображаемой линии, проходящей по потолку на равноудаленном расстоянии от длинных сторон коридора. Светильники должны быть направлены строго вертикальной вниз (по перпендикуляру к полу). При этом Вовочка считает, что необходимо разместить светильники так, чтобы пол коридора был полностью освещен прямым светом (в противном случае щенок Вовочки Ньютон будет прятать в темных углах тапки, которые нужно будет искать).

Вовочка хочет знать, какое минимальное количество светильников ему необходимо приобрести для освещения коридора согласно приведенным выше требованиям?

Входные данные

В единственной строке входного файла через пробел записаны 4-ре целых положительных числа ***A***, ***L***, ***W***, ***H***, где A – угол рассеивания светильников, заданный в градусах ($10 \le A \le 150$), а L, W и H – длина, ширина и высота (высота потолка) коридора соответственно, заданные в одинаковых единицах измерения ($1 \le L, W, H \le 10^5$, $W \le L$). Напомним, что коридор имеет строгую перпендикулярную форму.

Выходные данные

В единственной строке выходного файла надо записать одно число N_{min} – минимальное количество светильников, которые потребуются Вовочке для освещения коридора с соблюдением всех озвученных требований. Если это невозможно, то вместо N_{min} требуется записать число 0.

Пример входного файла (stdin / input.txt)	**Пример выходного файла (stdout / output.txt)**	**Комментарий**
60 1000 200 300	4	
30 50 20 30	0	Каждый потолочный светильник будет давать на полу круглое пятно света диаметром менее 17 единиц. Рядом с длинными стенами коридора будет оставаться темное пространство (ширина коридора – 20).

Решение жюри

```
#define _USE_MATH_DEFINES

#include <fstream>
#include <cmath>

using namespace std;


int main()
{
  ifstream cin("input.txt");
  ofstream cout("output.txt");

  int a, l, w, h;
  cin >> a >> l >> w >> h;

  int n = 0;

  double b = 2 * M_PI / 360 * a / 2;
  double r = h * tan(b);
  if (r * 2 > w) {
    double x = sqrt(4 * r * r - w * w);
    n = ceil(l / x);
  }

  cout << n << endl;
}
```

Задача 3. Красивые букеты

Входной файл	стандартный ввод / input.txt
Выходной файл	стандартный вывод / output.txt
Ограничение времени (сек/тест)	1
Ограничение памяти (Мб)	64

Условие задачи

В день 8-го Марта Вовочка решил поздравить как можно большее количество представительниц прекрасного пола (маму, сестру, одноклассниц и, конечно же, учителей). Но какое же поздравление в такой день без букета цветов! Поэтому накануне Вовочка пошел в ближайший цветочный магазин и скупил там все имеющиеся тюльпаны. Тюльпаны были разных цветов и теперь перед Вовочкой стоит задача сформировать *красивые* букеты. Вовочка считает, что *красивые* букеты состоят из нечетного количества, но не менее 3-х, тюльпанов различных цветов (т.е. цвета тюльпанов в букете не должны повторяться).

Помогите Вовочке сформировать наибольшее количество красивых букетов из имеющихся у него тюльпанов.

Входные данные

В первой строке входного файла записано целое число $\boldsymbol{N}$ – кол-во различных цветов купленных Вовочкой тюльпанов ($0 \le N \le 100$). Во второй строке через пробел записано N целых чисел $\boldsymbol{C_1}$, $\boldsymbol{C_2}$, ... , $\boldsymbol{C_N}$ – количество тюльпанов каждого цвета ($1 \le C_i \le 100$, $1 \le i \le N$).

Выходные данные

В первой строке входного файла надо вывести число $\boldsymbol{M}$ – максимальное кол-во красивых букетов, которые может составить Вовочка. Далее в M строках необходимо описать красивые букеты в виде набора чисел $\boldsymbol{F_{j1}}$, $\boldsymbol{F_{j2}}$, … , $\boldsymbol{F_{jK}}$ – номеров цветов тюльпанов, составляющих j-ый букет ($1 \le \boldsymbol{F_{jk}} \le$ N, $1 \le j \le M$, $1 \le k \le K$, $3 \le K$, K – нечетное).

В случае существования различных вариантов составления максимального кол-ва красивых букетов необходимо вывести любой из них.

Пример входного файла (stdin / input.txt)	**Пример выходного файла (stdout / output.txt)**	**Комментарий**
7 1 3 10 1 4 1 2	6 2 3 5 3 5 7 3 6 7 1 2 3 2 3 5 3 4 5	При этом у Вовочки еще останутся 4 тюльпана одного цвета – цвета 3, так как было опрометчиво куплено целых 10 тюльпанов данного цвета.
5 1 1 1 1 1	1 4 5 2	Любые 3 тюльпана образуют красивый букет.
5 2 3 2 2 2	3 1 2 3 3 4 5 1 2 5 4 2	По условию задачи букет из 5-ти различных тюльпанов также красивый :)
1 7	0	Все тюльпаны одного цвета, из них составить красивые букеты не получится.

Решение жюри

```
#include <fstream>
#include <vector>

using namespace std;


int main()
{
  ifstream cin("input.txt");
  ofstream cout("output.txt");

  int n;
  cin >> n;
  vector<int> c(n);
  for (int i = 0; i < n; i++)
    cin >> c[i];

  vector<vector<int> > bouquets;

  for (;;) {
    vector<int> b(3, -1);

    for (int i = 0; i < 3; i++) {
      int m = 0, mj;
      for (int j = 0; j < n; j++)
        if (j != b[0] && j != b[1] && j != b[2] &&
            c[j] > m) {
          m = c[j];
          mj = j;
        }

      if (m > 0) {
        b[i] = mj;
        c[mj]--;
      }
      else
        break;
    }

    if (b[2] >= 0)
      bouquets.push_back(b);
    else
      break;
  }

  cout << bouquets.size() << endl;
  for (int i = 0; i < bouquets.size(); i++)
    cout << (bouquets[i][0] + 1) << ' '
         << (bouquets[i][1] + 1) << ' '
         << (bouquets[i][2] + 1) << endl;
}
```

Задача 4. Сладкая мечта

Входной файл	стандартный ввод / input.txt
Выходной файл	стандартный вывод / output.txt
Ограничение времени (сек/тест)	1
Ограничение памяти (Мб)	64

Условие задачи

Вовочка – большой любитель шоколада. Он хотел бы кушать шоколад каждый день, однако плитки шоколада дарят Вовочке не так часто, а тратить свои деньги на сладости Вовочка не хочет, так как копит средства на новый компьютер.

Вовочка сильно расстраивается, если в какой-то день у него не окажется шоколада, если в предыдущий день он ел шоколад. Вовочка твердо решил сильно не расстраиваться на каникулах. Поэтому он будет откладывать подаренные шоколадки "на будущее", чтобы с какого-то дня и до конца каникул позволить себе каждый день съедать по одной плитке шоколада.

Вовочка составил список, когда и сколько шоколадок он ожидает в подарок. Теперь он просит Вас помочь определить самый ранний день, когда он, согласно составленному списку, сможет наконец осуществить свою мечту и кушать по одной плитке шоколада каждый день до конца каникул.

Плитку шоколада можно съедать уже в день подарка. Также известно, что в последний день каникул Вовочка обязательно получит шоколад в подарок.

Входные данные

В первой строке входного файла записано целое число $\boldsymbol{N}$ ($1 \le N \le 10^5$) – количество дней, когда Вовочке будут подарены шоколадки.

На каждой из последующих N строк записаны через пробел по два целых числа $\boldsymbol{T_i}$ и $\boldsymbol{C_i}$ ($1 \le Ci, Ti \le 10^9$) – номера дней и количество плиток шоколада, который подарят Вовочке в этот день. Все дни T_i различны и идут в порядке возрастания.

Выходные данные

В выходной файл требуется записать единственное число $\boldsymbol{T_{min}}$ ($1 \le T_{min} \le 10^9$) – минимальный день, с которого Вовочка может начать открывать по одной в день накопленные и поступающие шоколадки, не пропуская ни одного дня до конца каникул.

Пример входного файла (stdin / input.txt)	**Пример выходного файла (stdout / output.txt)**	**Комментарий**
3 2 3 3 1 7 4	3	Если начинать есть шоколад по одной плитке со 2-го дня, то на 6-ой день у Вовочки не останется ни одной шоколадки.
4 1 1 2 1 3 1 4 1	1	Вовочка каждый день получает в подарок одну плитку шоколада, которую можно тут же открывать и съедать.

Решение жюри

```
#include <fstream>
#include <vector>
#include <algorithm>
#include <utility>

using namespace std;


int main() {
  ifstream cin("input.txt");
  ofstream cout("output.txt");

  int n;
  cin >> n;

  vector<pair<int, int> > vec;
  for (int i = 0; i < n; i++) {
    int t, c;
    cin >> t >> c;
    vec.push_back(make_pair(t, c));
  }

  sort(vec.begin(), vec.end());
  int tmp = 0, prevsince = vec.rbegin()->first;
  for (int i = vec.size() - 1; i >= 0; i--) {
    int diff = vec[i].first + vec[i].second - prevsince;
    prevsince = vec[i].first;
    tmp += diff;
    if (tmp > 0)
      tmp = 0;
  }

  cout << (vec[0].first - tmp) << endl;
}
```

Задача 5. Номера домов

Входной файл	стандартный ввод / input.txt
Выходной файл	стандартный вывод / output.txt
Ограничение времени (сек/тест)	1
Ограничение памяти (Мб)	64

Условие задачи

Последние несколько лет улица, на которой живет Вовочка, массово, но хаотично застраивалась новыми домами. В итоге поиск по адресу номера дома превратился в сущий кошмар. Поэтому мэрия решила навести порядок и переназначить номера некоторым домам.

Требования к новой нумерации домов следующие:

- Номера домов – целые положительные числа $\leq 2 * 10^9$, не повторяются.
- На одной стороне улицы должны находится четные номера домов, на другой – нечетные. Четной можно сделать произвольную сторону улицы.
- При движении вдоль улицы номера домов должны либо монотонно возрастать с обеих сторон, либо монотонно убывать (но не обязательно идти последовательно).

Перенумерация домов доставит массу неудобств их жителям (переоформление документов и т.д.), поэтому мэрия в настоящий момент разрабатывает план новой нумерация, которая будет удовлетворять озвученным условиям, но затронет по возможности наименьшее кол-во домов.

Вовочка хочет предложить мэрии свой проект новой нумерации домов. Однако, чтобы оценить оптимальность своего проекта, ему необходимо знать, какое количество домов необходимо перенумеровать в самом оптимальном с точки зрения минимизации количества перенумеруемых домов проекте. Помогите Вовочке найти это минимальное количество.

Входные данные

В первой строке входного файла через пробел записаны два числа $\boldsymbol{N}$ и $\boldsymbol{M}$ ($1 \leq N, M \leq 3 * 10^3$).

Во второй строке, разделенные пробелами, следуют N различных чисел $\boldsymbol{A_i}$ ($1 \leq A_i \leq 10^9$), представляющие последовательность номеров домов по одной из сторон при движении вдоль по улице.

В третьей строке расположены M различных чисел $\boldsymbol{B_j}$ ($1 \leq B_j \leq 10^9$), представляющие последовательность номеров домов по другой стороне улицы, если двигаться в том же направлении, при котором были получены номера A_i.

Выходные данные

В единственной строке выходного файла надо записать число $\boldsymbol{K}$ – минимальное количество домов, которое необходимо перенумеровать, чтобы удовлетворить требований к новой нумерации.

Пример входного файла (stdin / input.txt)	**Пример выходного файла (stdout / output.txt)**	**Комментарий**
4 4 5 12 3 6 9 7 4 1	3	Возможный вариант нумерации: 18 12 8 6 9 7 3 1
5 4 1 3 7 9 15 2 6 8 10	0	Нумерация домов уже удовлетворяет требованиям.

Решение жюри

```
#include <fstream>
#include <vector>
#include <algorithm>

using namespace std;

#define fi(i, f, t) for (int i = (f); i < (t); i++)
#define fri(i, f, t) for (int i = (f); i >= (t); i--)
#define fa(it, c) for (auto (it) = (c).begin(); (it) != (c).end(); (it)++)
#define fra(it, c) for (auto (it) = (c).rbegin(); (it) != (c).rend(); (it)++)
#define vecits(obj) obj.begin(), obj.end()


int make_correct(const vector<int> &v) {
  int n = v.size();
  vector<int> dp(v);
  dp[0] = (v[0] == 0 ? 1 : 0);

  fi (i, 1, n) {
    int bestv = n + 1;
    if (v[i] >= (i + 1))
      bestv = i;

    fi (j, 0, i)
      if (v[j] <= (v[i] - (i-j)) && v[j] > 0)
        bestv = min(bestv, i - j - 1 + dp[j]);

    dp[i] = bestv;
  }

  int res = n;
  fi (i, 0, n)
    res = min(res, n - i - 1 + dp[i]);

  return res;
}


int main() {
  ifstream cin("input.txt");
  ofstream cout("output.txt");

  int n,m;
  cin >> n >> m;

  vector<int> vecs[2];
  fi (i, 0, n + m) {
    int tmp;
     cin >> tmp;
     if (i < n)
       vecs[0].push_back(tmp);
     else
       vecs[1].push_back(tmp);
  }

  int best_res = n + m;

  fi (dir, 0, 2) {
    fi (odd, 0, 2) {
      vector<int> oddv, evenv;
      fa (it, vecs[odd])
        if (*it % 2 == 1)
          oddv.push_back(*it / 2 + 1);
        else
          oddv.push_back(0);
```

```
      fa (it, vecs[1 - odd])
        if (*it % 2 == 0)
          evenv.push_back(*it/2);
        else
          evenv.push_back(0);

      best_res = min(best_res, make_correct(oddv) + make_correct(evenv));
    }

    reverse(vecits(vecs[0]));
    reverse(vecits(vecs[1]));
  }

  cout << best_res << endl;
}
```

Задача 6. Снежинки

Входной файл	стандартный ввод / input.txt
Выходной файл	стандартный вывод / output.txt
Ограничение времени (сек/тест)	1
Ограничение памяти (Мб)	64

Условие задачи

На новогодних каникулах Вовочка, запустив непроверенную программу, скаченную из Интернета, заразил свой компьютер вирусом «Дед Мороз». С вирусом он справился, однако остались последствия. «Дед Мороз» оказался большим шутником и зашифровал весь исходный код проектов, с которыми Вовочка работал. Зашифрованные файлы остались текстовыми, но их содержимое на первый взгляд превратилось в случайный набор символов.

Кроме того, Вовочка заметил, что в этих файлах часто повторяется шаблон, похожий на нарисованную псевдографикой снежинку. «Снежинки» встречаются разного размера. Граница «снежинки» формируются тремя символами: '|' – вертикальная черта, '–' – знак «минус», '*' – «звездочка», тело «снежинки» состоит из символов '.' – точка (в зашифрованных файлах вообще все пробелы заменены точками). На рисунке справа показаны, как выглядят две «снежинки» – размера 1 и размера 2. (формально размер «снежинки» можно определить как $M/4$, где M – кол-во «звездочек» во внутренних углах фигуры). «Снежинки» большего размера формируются аналогично. Для наглядности на рисунке в виде сетки обозначены границы символов, т.е. каждая клеточка – один символ. Выделенные цветом клеточки – символы, образующие границу «снежинки».

```
. . . . . . . . . . . . . . . . . . . . .
1 ) . * – * . . . . . . . . * – * . . . . .
. . . | . | . . . . 2 ) . . | . | . . . . .
. * – * . * – * . . . . * – * . * – * . . .
. | . . . . . | . . . . | . . . . . | . . .
. * – * . * – * . . * – * . . . . . * – * .
. . . | . | . . . . | . . . . . | . . . |
. . . * – * . . . . * – * . . . . . * – * .
. . . . . . . . . . . | . . . . . | . . .
. . . . . . . . . . . * – * . * – * . . .
. . . . . . . . . . . . . | . | . . . . .
. . . . . . . . . . . . . * – * . . . . .
. . . . . . . . . . . . . . . . . . . . .
```

В интернете Вовочка прочитал, что зашифрованные вирусом «Дед Мороз» файлы можно расшифровать по специальному алгоритму, ключом к которому являются положение и размер всех «снежинок», встречающихся в файле. Помогите Вовочке найти этот ключ.

Входные данные

Входной файл состоит из произвольного количества строк, но не более 100. Строки могут содержать буквы латинского алфавита, цифры, любые знаки препинания и спецсимволы (а более конкретно символы из набора "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890.%!@#$^&*(){}[],.;:"). Длина каждой строки не превышает 100 символов, строки в одном файле могут быть разной длины, но не пустые. Последняя строка файла содержит единственный символ – точку.

Выходные данные

В первой строке входного файла нужно записать единственное число $\boldsymbol{N}$ – количество найденных «снежинок».

В последующих N строках надо записать информацию о каждой найденной «снежинке» в виде разделенных пробелами трех чисел $\boldsymbol{R_i}$, $\boldsymbol{C_i}$, $\boldsymbol{S_i}$, где R_i, C_i – позиция центра «снежинки» (R_i – номер строки, C_i – номер символа в строке, нумеруются с 1-цы, см. пример входных-выходных данных), а S_i – размер i-той «снежинки» (см. определение выше).

Две «снежинки» могут иметь общую границу (часть границы). Также напомним, что если внутри тела «снежинки» встречаются любые символы, кроме точек, то такая фигура «снежинкой» не является. Также не существует «снежинок» размера 0.

Список обнаруженных «снежинок» можно выводить в произвольном порядке.

Пример входного файла (stdin / input.txt)	**Пример выходного файла (stdout / output.txt)**	**Комментарий**
..... 1).*-*.....*-*..... ...|.|....2)..|.|..... .*-*.*-*....*-*.*-*... .|.....|....|.....|... .*-*.*-*..*-*.....*-*. ...|.|....|.....|. ...*-*....*-*.....*-*. .....|.....|... .....*-*.*-*... .....|.|..... .....*-*.....	2 5 5 1 7 16 2	Пример соответствует рисунку в условии задачи.
...*-*...*-*..... .av|.|web|.|...eihdik.. .*-*.*-*-*.*-*....we .|.....|.....|... .*-*.*-*.....*-*. ...|.|.....|. .2.*-*-*.....*-*.2 .....|.|.....|.|..2 dwe*-*.*-*.*-*.*-*.. ...|.....|.|.....| .*-*.....*-*-*.*-* .|.....|.|.|.337greh% -*.....*-*-*...34r .....|.....r3 -*.....*-* .|.....|..wed .*-*.....*-*.....we%1() 5..|.....|23wed....^781 278*-*.*-*dghw23 .....|.|.....*-* weghd*-*.....45...|.| .....23..67....*-*.*-* bvgh.%^.we.....|.....| .....3.*-*.not.*-* HB@bshk4$.....|snowflake| ..wellkwef.34.*-*..!..*-* 3457856.....|.....| ..5345.45.65....*-*.*-* 5.45.345.....|.| ....34587.9879....*-* 42re.hegjhb werkjkwernkjn3urn3untiunu3 # # end. .	3 4 5 1 10 15 1 6 11 2	«Снежинка» размером 3 с центром (14, 7) своим левым краем чуть-чуть «вылезает» за границу файла, поэтому формально снежинкой не считается. Фигура с центром (25, 20) в нижнем правом углу файла также не является «снежинкой» так содержит внутри себя символы, отличные от точек. Таким образом остаются три «правильных» «снежинки». При этом большая из 3-х «снежинок» граничит (имеет общую границы) с двумя меньшими.

class.NewClass.{
}
.
	0	«Снежинок» нет.

Решение жюри

```
#include <fstream>
#include <sstream>
#include <vector>
#include <string>

using namespace std;


char ch(vector<string> &f, int r, int c) {
  return !(0 <= r && r < f.size()) ||
         !(0 <= c && c < f[r].size()) ? '!' : f[r][c];
}


bool check(vector<string> &f, int r, int c, int s) {
  for (int h = 3 + s * 4, w = 3; h > 0; h -= 4, w += 4) {
    int r1 = r - w / 2, r2 = r + w / 2,
        c1 = c - h / 2, c2 = c + h / 2;
    if (!(ch(f, r1, c1) == '*' &&
          ch(f, r1, c1 + 1) == '-' &&
          ch(f, r1, c1 + 2) == '*'
          &&
          ch(f, r1, c2) == '*' &&
          ch(f, r1, c2 - 1) == '-' &&
          ch(f, r1, c2 - 2) == '*'
          &&
          ch(f, r2, c1) == '*' &&
          ch(f, r2, c1 + 1) == '-' &&
          ch(f, r2, c1 + 2) == '*'
          &&
          ch(f, r2, c2) == '*' &&
          ch(f, r2, c2 - 1) == '-' &&
          ch(f, r2, c2 - 2) == '*'
          &&
          //ch(f, r1, c1) == '*' &&
          ch(f, r1 + 1, c1) == '|' &&
          ch(f, r1 + 2, c1) == '*'
          &&
          //ch(f, r2, c1) == '*' &&
          ch(f, r2 - 1, c1) == '|' &&
          ch(f, r2 - 2, c1) == '*'
          &&
          //ch(f, r1, c2) == '*' &&
          ch(f, r1 + 1, c2) == '|' &&
          ch(f, r1 + 2, c2) == '*'
          &&
          //ch(f, r2, c2) == '*' &&
          ch(f, r2 - 1, c2) == '|' &&
          ch(f, r2 - 2, c2) == '*'))
      return false;

    int c1_prev = c1 + 2,
        c2_prev = c2 - 2;
    for (int cc = c1 + 1; cc < c2; cc++) {
      if (c1_prev < c1_prev &&
          c1_prev < cc && cc < c2_prev)
        continue;
      for (int rr = r1 + 1; rr < r2; rr++)
        if (ch(f, rr, cc) != '.')
          return false;
    }
  }

  return true;
}
```

```
int main()
{
  ifstream cin("input.txt");
  ofstream cout("output.txt");

  vector<string> file;
  while (!cin.eof()) {
    string line;
    getline(cin, line);
    file.push_back(line);
  }

  stringstream ss;
  int n = 0;
  for (int r = 0; r < file.size(); r++)
    for (int c = 0; c < file[r].size(); c++)
      for (int s = 25; s > 0; s--)
        if (check(file, r, c, s)) {
          n++;
          ss << r + 1 << ' ' << c + 1 << ' ' << s << endl;
        }

  cout << n << endl << ss.str();
}
```