Задача 1. Друзья и учеба

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

Условие задачи

На прошлой неделе, когда весь класс писал тест по математике, Вовочка болел. Директор школы требует, чтобы данный тест был написан всеми учениками, поэтому завтра этот тест предстоит решать Вовочке. Для Вовочки результат данного теста ни на что не влияет, т.к. он в этом году занял призовое место на областной олимпиаде по математике и директор лично перед всем классом пообещал поставить Вовочке по математике пять в году.

Тест оценивается в баллах от 0 до некоторого максимально возможного количества баллов $\boldsymbol{M_{max}}$ с шагом в один балл. Вовочка отлично разбирается в математике и поэтому уверен, что может набрать в данном тесте максимально возможное количество баллов (соответственно, и любое другое количество баллов по своему желанию). Однако Вовочка опасается, что если его результат будет сильно отличаться от результатов других учеников, то последние перестанут с ним дружить. Если говорить более строго, Вовочка считает, что с ним перестанут дружить те ученики в классе, результат которых будет отличаться от результата Вовочки более, чем на $\boldsymbol{K}$ баллов в любую сторону.

Помогите Вовочке определить, сможет ли он написать тест так, чтобы сохранить всех своих друзей (в данный момент все ученики в классе – друзья Вовочки).

Входные данные

В первой строке через пробел записаны три целые числа $\boldsymbol{N}$, M_{max} и K ($1 \le N, M_{max} \le 100$, $0 \le K \le 100$) – количество остальных учеников в классе, исключая Вовочку, максимальный балл, который можно набрать по тесту, правильно ответив на все вопросы и максимально допустимая разница в баллах с кем-то из одноклассников, чтобы последний остался другом Вовочки.

В следующей строке через пробел записаны N целых чисел $\boldsymbol{M_i}$ ($0 \le M_i \le M_{max}$) – баллы по тесту, набранные одноклассниками Вовочки.

Выходные данные

Выведите строку "yes" (без кавычек), если Вовочка сможет написать тест, сохранив всех своих друзей, "no" (также без кавычек) – в противном случае.

Пример входного файла (input.txt)	**Пример выходного файла (output.txt)**	**Комментарий**
7 10 3 3 5 8 5 8 7 4	yes	Чтобы сохранить всех своих друзей, Вовочка может написать тест на 5 или на 6 баллов.
3 20 5 15 3 10	no	

Решение

Очевидно, что оптимальной стратегией для Вовочки будет написать тест на средний балл между минимальным и максимальным баллом среди результатов его друзей. При этом Вовочка сможет сохранить всех друзей, если:

$$\max(\boldsymbol{M_i}) - \min(\boldsymbol{M_i}) \leq 2K$$

Решение жюри

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

using namespace std;


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

  int n, k;
  cin >> n >> k >> k;

  int m_min, m_max;
  for (int i = 0; i < n; i++) {
    int mi;
    cin >> mi;
    if (i == 0)
      m_min = m_max = mi;
    m_min = min(m_min, mi);
    m_max = max(m_max, mi);
  }

  cout << (m_max - m_min <= 2 * k ? "yes" : "no") << endl;
}
```

Задача 2. Система уравнений

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

Условие задачи

Учитель математики составляет индивидуальные задания на контрольную для учеников седьмого класса. В качестве одного из заданий должна быть система, состоящая из двух уравнений, вида:

$$ax + by = c$$
$$dx + ey = f$$

Вместо букв ***a***, ***b***, ***c***, ***d***, ***e***, ***f*** учитель хочет подставить числа так, чтобы единственным решением системы были целые числа x и y ($-99999 \le x, y \le 99999$). Чтобы ученикам не было слишком легко, требуется, чтобы все шесть коэффициентов были ненулевыми. Так как учитель не желает пугать учеников длинными числами, дополнительно он хочет, чтобы коэффициенты состояли не более, чем из пяти цифр ($-99999 \le a, b, c, d, e, f \le 99999$). Учитель просит Вовочку разработать компьютерную программу для составления систем уравнений согласно заданным требованиям.

Входные данные

Единственная строка содержит два целых числа, разделенных пробелом – x и y, единственное решение системы уравнений.

Выходные данные

Если невозможно составить систему уравнений, удовлетворяющую заданным требованиям, выведите единственное слово "no" (без кавычек).

Если можно составить систему уравнений, удовлетворяющую заданным требованиям, в первой строке выведите слово "yes" (без кавычек). Во второй строке через пробел выведите три целых числа – коэффициенты первого уравнения a, b и c, а в третьей строке аналогичным образом – коэффициенты второго уравнения d, e и f.

Пример входного файла (input.txt)	**Пример выходного файла (output.txt)**
1 5	yes 10 -1 5 20 30 170
0 99999	yes -3 1 99999 5 -1 -99999
0 0	no

Решение

Во-первых, надо сразу учесть, что для ($x = 0$, $y = 0$) решения задачи не существует.

Во-вторых, также надо сразу учесть, что система будет иметь единственное решение, только если $a / b \neq d / e$, или, чтобы не беспокоиться о делении на ноль, $ae \neq bd$.

Рассматривая граничные случаи можно легко показать, что для любых x и y, удовлетворяющих условию задачи, можно подобрать a, b, d, e в диапазоне от −2 до 2, такие, что c и f будут попадать в требуемый диапазон (от −99999 до 99999) и $ae \neq bd$:

1) для ($x = 99999$, $y = 99999$) подойдут, например, ($a = 2$, $b = -1$, $d = -1$, $e = 2$)

2) для ($x = 99999$, $0 \leq y < 99999$) подойдут, например, ($a = 1$, $b = -2$, $d = -1$, $e = 1$)

3) если $|x|$ и $|y|$ сильно меньше 99999, то и подавно существование подходящих a, b, d, e в диапазоне от −2 до 2 очевидно

Те же самые случаи с другими знаками при x, y просто приведут к другим знакам у a, b, d и e. Если (x, y) поменять местами, то поменяются местами (a, b) и (d, e).

Таким образом, чтобы не рассматривать скрупулезно все случаи, достаточно перебрать все комбинации a, b, d, e в в диапазоне от −2 до 2 (на всякий случай расширить диапазон от −5 до 5 – в этом случае можно даже найти подходящие a, b, c, d, e, f, не равные друг другу), для каждой комбинации вычислить c и f и проверить, что a, b, c, d, e, f нам подходят.

Решение жюри

```
#include <fstream>

using namespace std;


bool check(int a, int b, int c, int d, int e, int f) {
  int nums[6];
  nums[0] = a;
  nums[1] = b;
  nums[2] = c;
  nums[3] = d;
  nums[4] = e;
  nums[5] = f;

  if (a * e == b * d)
    return false;

  for (int i = 0; i < 6; i++)
    if (nums[i] < -99999 || nums[i] > 99999 || nums[i] == 0)
      return false;

  return true;
}


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

  int x, y;
  cin >> x >> y;

  if (x == 0 && y == 0) {
    cout << "no" << endl;
    return 0;
  }

  for (int a = -5; a <= 5; a++)
    for (int b = -5; b <= 5; b++)
      for (int d = -5; d <= 5; d++)
```

```
        for (int e = -5; e <= 5; e++) {
          int c = a * x + b * y,
              f = d * x + e * y;
          if (check(a, b, c, d, e, f)) {
            cout << "yes" << endl;
            cout << a << ' ' << b << ' ' << c << endl;
            cout << d << ' ' << e << ' ' << f << endl;
            return 0;
          }
        }

  cout << "no" << endl;
}
```

Задача 3. Матрешки

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

Условие задачи

Вовочка не только хорошо учится, но и участвует в школьной художественной самодеятельности. Недавно, на фестивале русской народной культуры, в качестве сувенира Вовочке вручили набор из четырех матрешек ручной работы. Рассматривая сувенир, Вовочка придумал оригинальную задач для участников школьного кружка по информатике. Вовочка считает, что знает решение данной задачи, но хочет проверить себя. Для этого необходимо, чтобы кто-то еще решил данную задачу для сравнения ответов, и желательно, чтобы это был не один человек.

Итак условие задачи: четыре матрешки разного размера стоят в один ряд и необходимо как можно быстрее собрать их в одну (вложить друг в друга). Скорость сборки оценивается в количестве *простых* действий. Возможны все два простых действия:

- поменять местами соседние матрешки;
- меньшую соседнюю матрешку вложить в большую.

Если в какой-то матрешке A уже вложена меньшая матрешка B, то ни в матрешку A (между A и B), ни в матрешку B вложить некую третью матрешку C уже нельзя (это не будет простым действием, т.к. в какой-то момент такой операции надо будет одновременно работать с тремя матрешками, не вложенными друг в друга). Но при этом матрешку A (с вложенными в нее матрешками), можно вкладывать в матрешку C (естественно, если матрешка C больше матрешки A).

Входные данные

В единственной строке через пробел записаны целые числа $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ ($1 \le A, B, C, D, \le 99$) – размеры матрешек в порядке их расположения в ряду слева направо.

Выходные данные

Выведите единственное число $\boldsymbol{N_{min}}$ – минимальное количество простых действий, которые приведут к сбору данного набора матрешек с учетом их первоначального взаимного расположения.

Пример входного файла (input.txt)	**Пример выходного файла (output.txt)**	**Комментарий**
3 7 2 4	4	Возможная последовательность действий (номер действия: взаимное расположение оставшихся матрешек после данного действия): 1: 7 3 2 4 2: 7 3 4 3: 7 4 4: 7
7 3 5 9	3	1: 7 5 9 2: 7 9 3: 9

Решение

Очевидно, что сначала мы должны поместить самую мелкую матрешку в матрешку следующую по размеры (для этого данные матрешки должны стоять рядом) и т.д.

Обозначим матрешки буквами *a*, *b*, *c* и *d* в порядке увеличения размера (a – самая маленькая). Возможных комбинаций букв будет 4! = 24. Можно даже для каждой комбинации в уме посчитать кол-во требуемых операции и забить в код программы в виде словаря – и это уже будет решением.

В процессе такого подсчета очевидным станет алгоритм. Запишем в массив int pos[4] положение букв *a*, *b*, *c*, *d*. В pos[0] будет позиция (индекс) буквы *a*, pos[1] – *b*, в pos[2] – *c* и в pos[3] – *d*.

Если расстояние между *a* и *b* равно 2 (abs(pos[0] – pos[1]) = 3), т.е. самые мелкие матрешки стоят в самой левой и правой позиции, то ответ = 5 (2 действия, чтобы их сблизить в середине ряда и 3 действия, чтобы поместить друг в друга).

Если расстояние между *a* и *b* равно 1 (abs(pos[0] – pos[1]) = 2), то ответ 4 (1 действие, чтобы их сблизить в середине ряда и 3 действия, чтобы поместить друг в друга).

Если *a* и *b* рядом (abs(pos[0] – pos[1]) = 1), то ответ 3 или 4 в зависимости от того, есть ли рядом с *a* или *b* (после первого действия останется только *b*) матрешка *c*.

Решение жюри

```
#include <fstream>
#include <map>
#include <cmath>

using namespace std;

/*
  a b c d -> 3
  a b d c -> 4
  a c b d -> 3
  a c d b -> 5
  a d b c -> 4
  a d c b -> 5
  b a c d -> 3
  b a d c -> 4
  b c a d -> 4
  b c d a -> 5
  b d a c -> 4
  b d c a -> 5
  c a b d -> 3
  c a d b -> 4
  c b a d -> 3
  c b d a -> 4
  c d a b -> 4
  c d b a -> 4
  d a b c -> 3
  d a c b -> 4
  d b a c -> 3
  d b c a -> 4
  d c a b -> 3
  d c b a -> 3
*/


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

  int nums[4];
  for (int i = 0; i < 4; i++)
    cin >> nums[i];
```

```
  map<int, int> m;
  for (int i = 0; i < 4; i++)
    m[nums[i]] = i;
  int pos[4], idx = 0;
  for (map<int, int>::iterator it = m.begin(); it != m.end(); it++, idx++)
    pos[idx] = it->second;

  int result;
  if (abs(pos[0] - pos[1]) == 3)
    result = 5;
  else if (abs(pos[0] - pos[1]) == 2)
    result = 4;
  else
    if (((pos[0] == 0 || pos[1] == 0) && pos[2] == 3) ||
        ((pos[0] == 3 || pos[1] == 3) && pos[2] == 0))
      result = 4;
    else
      result = 3;

  cout << result << endl;
}
```

Задача 4. Проектирование моста

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

Условие задачи

Для закрепления знаний по теме «Оптимизация» на дополнительных занятия по информатике учитель раздал ученикам индивидуальные практические задания. Вовочке досталась задача о проектированием моста.

На прямой линии между берегами, вдоль которой будет проходить мост, существует $\boldsymbol{N}$ ($3 \le N \le 1000$) равномерно распределенных точек (т.е. точки расположены на одинаковом расстоянии $\boldsymbol{D}$ друг от друга). В каждой из этих точек был проведен замер глубины реки. Мост строится на некотором количестве опор, каждая из опор должна находиться в одной из точек с измеренной глубиной. Для повышения надежности моста максимально допустимое расстояние между двумя соседними опорами – $3D$ (другими словами, между соседними опорами может быть максимум две пропущенных точки). Также опоры обязательно должны быть размещены рядом с берегами реки – в первой и последней точке.

Известно, что глубина реки вдоль линии моста сначала с каждой точкой увеличивается, а затем уменьшается. Требуется спроектировать мост таким образом, чтобы суммарная длина $\boldsymbol{L}$ опор моста (другими словами, суммарное расстояние от дна до поверхности реки в точках с опорами) была минимально возможной. Помогите Вовочке определить величину L.

Входные данные

Первая строка содержит целое число N ($3 \le N \le 1000$). Во второй строке через пробел записано N целых чисел $\boldsymbol{A_i}$ ($1 \le A_i \le 1000$) – результат измерения глубины в точке i. Значение A_i в строке сначала увеличивается, затем уменьшается.

Выходные данные

Выведите одно число L – суммарное расстояние от дна до поверхности реки в точках с опорами при оптимальном размещении опор.

Пример входного файла (input.txt)	**Пример выходного файла (output.txt)**
6 1 3 4 7 3 2	7

Решение

Будем рассуждать следующим образом. Найдем самую глубокую точку. Очевидно, что от этой точки влево и вправо мы должны расставлять опоры как можно дальше друг от друга (но не забыть поставить опору в первой и последней точке). Очевидно, что возможно это не будет оптимальным решением и возможно опору нам надо поставить слева или справа от самой глубокой точки. Но при этом остальные опоры мы должны все равно расставлять как можно дальше друг от друга. Таким образом, за исключением первой и последней опоры (соответственно в первой и последней точке) остальные опоры всегда стоят друг от друга на максимальном расстоянии друг от друга (через две пустые точки). Таким образом надо посчитать суммы глубин опор (за исключением первой и последней) по описанному выше принципу, начиная со 2-ой опоры, с 3-ей и 4-ой. Ответом будет минимальная сумма + длина первой и последней опоры.

Для решения задачи можно использовать и метод динамического программирования (такое решение будет работать даже в случае совершенно произвольного чередования глубин). Обозначим за dp[i] – минимальную сумму опор, включая i-ую, от левой границы моста. Очевидно, что dp[i] = min(dp[i – 1], dp[i – 2], dp[i – 3]) + a[i]. Ответом будет dp[n]. Единственно учесть, чтобы при вычислениях не выйти за левую границу массива dp.

Решение жюри

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

using namespace std;


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

  int n;
  vector<int> a;

  cin >> n;
  for (int i = 0; i < n ; i++) {
    int temp;
    cin >> temp;
    a.push_back(temp);
  }

  int result = 0;
  for (int s = 1; s <= 3; s++) {
    int sum = a[0];
    for (int i = s; i < n - 1; i += 3)
      sum += a[i];
    sum += a[n - 1];
    result = result ? min(result, sum) : sum;
  }

  cout << result << endl;
}
```

Второй вариант решения методом динамического программирования (это решение будет работать даже в случае совершенно произвольного чередования глубин):

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

using namespace std;
```

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

  int n;
  vector<int> a;

  cin >> n;
  for (int i = 0; i < n ; i++) {
    int temp;
    cin >> temp;
    a.push_back(temp);
  }

  vector<int> dp(n);
  dp[0] = a[0];
  for (int i = 1; i < n; i++) {
    dp[i] = dp[i - 1] + a[i];
    if (i > 1)
      dp[i] = min(dp[i], dp[i - 2] + a[i]);
    if (i > 2)
      dp[i] = min(dp[i], dp[i - 3] + a[i]);
  }

  cout << dp[n - 1] << endl;
}
```

Задача 5. Инвестиции

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

Условие задачи

Вовочка очень любит конфеты кондитерской фабрики, которая находится в паре кварталов от его дома. Недавно Вовочка узнал, что он может стать ее совладельцем, скупая акции кондитерской фабрики на фондовом рынке. Счастью Вовочки не было предела!

Вовочка твердо решил, что каждый день он будет экономить $\boldsymbol{M}$ рублей и покупать на них максимально возможное количество акций компании. Можно купить или продать только целое число акций. Вовочка будет действовать по простому алгоритму: каждый день он будет покупать максимальное число акций на все доступные ему в этот день деньги, сумма которых будет составлять сумму, оставшуюся с прошлого дня, и M рублей, сэкономленных в данный день. При этом Вовочка никогда не будет продавать купленные акции и на что-либо еще тратить деньги, оставшиеся в какой-либо день после покупки акций.

Прошло $\boldsymbol{N}$ дней и Вовочка хочет подсчитать два числа: сколько акций он купил и сколько максимально акций он мог бы купить, если бы действовал оптимально. Под оптимальностью подразумевается, что Вовочка бы совершал только такие покупки и продажи акций, которые в итоге привели бы его к владению максимальным количеством акций по прошествии N дней.

Для каждого из N дней известен курс акций на фондовом рынке – $\boldsymbol{P_i}$ рублей за одну акцию. Курс покупки не отличается от курса продажи.

Входные данные

В первой строке через пробел записаны два целых числа: N ($1 \le N \le 10^5$) – количество прошедших дней и M ($1 \le M \le 10^6$) – сумма рублей, которые Вовочка экономил каждый день.

Во второй строке записаны N целых чисел P_i ($1 \le P_i \le 10^6$) – стоимость покупки (продажи) акций в каждый из N дней.

Выходные данные

В единственной строке через пробел выведите два числа $\boldsymbol{X}$ и $\boldsymbol{X_{max}}$ – фактическое количество купленных акций и максимальное количество акций, которыми Вовочка мог бы владеть по прошествии N дней при оптимальной стратегии.

Гарантируется, что при любой стратегии ни в какой из N дней у Вовочки не могла оказаться на руках сумма, большая $2 * 10^9$ рублей.

Пример входного файла (input.txt)	**Пример выходного файла (output.txt)**	**Комментарий**
3 100 10 100 4	36 300	Оптимально в первый день на сэкономленные деньги купить 10 акций, продать их во второй день за 1000 рублей и на третий день за 1200 рублей (с учетом продажи и сэкономленных денег во второй и третий день) купить 300 акций
5 100 5 1 3 4 2	228 566	

Решение

Рассмотрим, как нужно максимизировать кол-во акций. Если для какого-то очередого i-го дня в следующий (i+1)-ый день цена акций будет ниже, то надо в текущий день имеющиеся акции продавать, т.к. уже в следующий день мы потенциально сможем купить больше акций (ну или то же кол-во, но денег у нас будет больше). Если цена на следующий день выше, то акции в этот день нужно покупать, т.к. уже завтра эти акции можно продать, заработав дополнительные деньги. В последний день на все имеющиеся деньги покупаем акции.

Второй вариант решения задачи возможен методом динамического программирования. Будем понимать под нашим положением пару чисел – кол-во у нас акций и кол-во денег (можно описать в виде структуры данных). Будем для каждого следующего дня подсчитывать два лучших положения – с точки зрения максимизации кол-ва акций и максимизации денег. При этом надо учесть, что каждое оптимальное положение в текущий день может быть получено как из оптимального положения в предыдущий день по кол-ву акции, так и из оптимального положения в предыдущий день по кол-ву денег (но если максимальное кол-во акций в двух случаях одинаково, то надо максимизировать кол-во денег). Данный алгоритм наглядно описывает второй пример решения жюри (названия переменных говорят сами за себя).

Решение жюри

```
#include <fstream>

using namespace std;


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

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

  int shares0 = 0, money0 = 0,
      shares = 0, money = 0;

  int p, p_next;
  cin >> p;
  for (int i = 0; i < n - 1; i++) {
    cin >> p_next;

    shares0 += (money0 + m) / p;
    money0 = (money0 + m) % p;

    money += m;
    if (p > p_next) {
      money += shares * p;
      shares = 0;
    }
    else {
      shares += money / p;
      money %= p;
    }

    p = p_next;
  }
  shares0 += (money0 + m) / p;
  //money0 = (money0 + m) % p;
  shares += (money + m) / p;
  //money += (money + m) % p;

  cout << shares0 << ' ' << shares << endl;
}
```

Второй вариант решения методом динамического программирования:

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

using namespace std;


struct state_t {
  int shares;
  int money;

  bool operator <(const state_t& oth) const {
    return shares == oth.shares ? money < oth.money : shares < oth.shares;
  }
};


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

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

  state_t
    only_buying = { 0, 0 },
    max_shares  = { 0, 0 },
    max_money   = { 0, 0 };

  for (int i = 0; i < n; i++) {
    int p;
    cin >> p;

    only_buying.shares = (only_buying.money + m) / p;
    only_buying.money = (only_buying.money + m) % p;

    state_t
      max_money_from_max_shares = {
        0,
        max_shares.money + max_shares.shares * p + m
      },
      max_money_from_max_money = {
        0,
        max_money.money + m
      },
      max_shares_from_max_shares = {
        max_shares.shares + (max_shares.money + m) / p,
        (max_shares.money + m) % p
      },
      max_shares_from_max_money = {
        (max_money.money + m) / p,
        (max_money.money + m) % p,
      };
    max_money = max(max_money_from_max_shares, max_money_from_max_money);
    max_shares = max(max_shares_from_max_shares, max_shares_from_max_money);
  }

  cout << only_buying.shares << ' ' << max_shares.shares << endl;
}
```

Задача 6. Треугольник Серпинского

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

Условие задачи

На прошлой неделе на дополнительном занятии по математике Вовочка узнал про фракталы. Если говорить простыми словами (насколько это возможно в одном предложении), фракталы – это объекты, структура которых повторяется в составных частях объекта и так до бесконечности. Некоторые объекты в природе, а также математические объекты, обладают свойствами фракталов.

В качестве наглядных примеров учитель показал классу несколько примеров фракталов, построенных на плоскости. Среди этих примеров был треугольник Серпинского, на рисунке ниже показан принцип построения данной фрактала.

Вовочке идея фракталов очень понравилась, так что он в качестве тренировки решил написать программы для построения нескольких фракталов – первым он хочет нарисовать как раз треугольник Серпинского. Однако, вот незадача, компьютерную графику на уроках информатики еще не проходили, поэтому Вовочка решил «рисовать» некий аналог рассматриваемого фрактала с помощью псевдографики, т.е. текстовых символов, имитирующих графические примитивы.

Входные данные

Единственная строка содержит одно целое число $\boldsymbol{N}$ ($1 \le N \le 10$) – размерность «рисуемой» фигуры.

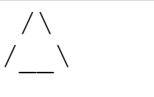
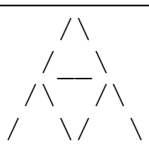
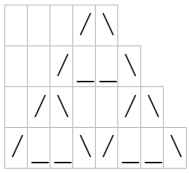
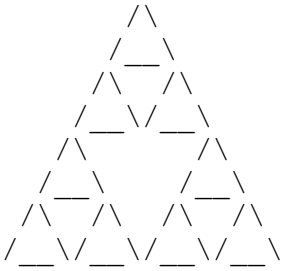
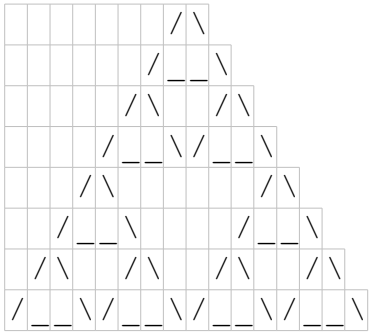
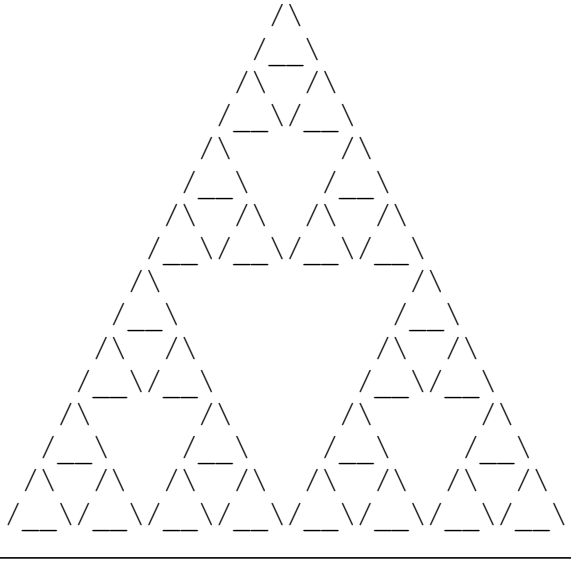
Выходные данные

С помощью символов псевдографики и пробелов должна быть построена фигура, напоминающая треугольник Серпинского. Допустимы только следующие символы:

- косая черта (слеш) – '/', ASCII-код символа 47 (0x2F);
- обратная косая черта (бекслеш) – '\', ASCII-код символа 92 (0x5C);
- подчеркивание – '_', ASCII-код символа 95 (0x5F);
- символы перевода строк.

Правила построения фигур, а также понятие их размерности, лучше всего иллюстрируются приведенными ниже примерами входных и выходных данных для N = 1, 2, 3 и 4.

Построенный треугольник должен касаться верхней и левой границы «рисунка». Каждая строка должна заканчиваться обратной косой чертой, без пробелов после, т.е. все строки будут разной длины. После последнего «графического» символа (обратной косой черты), должен быть перевод строки. Другими словами, правильный ответ в данной задаче должен полностью посимвольно совпадать с эталонным решением жюри.

Пример входного файла (input.txt)	**Пример выходного файла (output.txt)**	**Комментарий**
1		 Для наглядности в виде сетки обозначены границы символов, т.е. каждая клеточка – один символ (пустые клетки – пробелы, символы перевода строк не показаны)
2		
3		
4		

Решение

Рассмотрим любой треугольник из данного фрактала. Ширина H каждого такого треугольника (если считать по нижним клеткам) в 2 раза больше высоты W. При этом зависимость от N следующая: $H(N) = 2^N$, $W(N) = 2 * H(N)$

Прежде всего необходимо завести двумерный массив под формируемую «картинку» требуемого размера. Далее необходимо реализовать рекурсивную процедуру (в приведенном ниже коде функция part), которая умеет «рисовать» треугольник размера N (level) в прямоугольнике с верхним левым углом (left, top). Для $N = 1$ будет отдельная реализация, для $N > 1$ – рекурсивная, реализованная через три вызова данной процедуры для $N - 1$ и рассчитанных верхних левых углом соответствующих составных треугольников.

Ниже приводится решение жюри, которое достаточно очевидно.

Решение жюри

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

using namespace std;


void part(int level, int top, int left, vector<string> &field) {
  if (level == 1) {
    field[top][left] = field[top + 1][left - 1] = '/';
    field[top + 1][left - 1] = '/';
    field[top][left + 1] = field[top + 1][left + 2] = '\\';
    field[top + 1][left] = field[top + 1][left + 1] = '_';
  }
  else {
    level--;
    int partSize = 1 << level;
    part(level, top, left, field);
    part(level, top + partSize, left - partSize, field);
    part(level, top + partSize, left + partSize, field);
  }
}


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

  int level;
  cin >> level;

  int h = 1 << level,
      w = h << 1;
  vector<string> field(h, string(w, ' '));

  part(level, 0, h - 1, field);

  for (int i = 0; i < h; i++) {
    string line = field[i].substr(0, field[i].find_last_of('\\') + 1);
    cout << line.c_str() << endl;
  }
}
```