

Задача 1. Самое длинное слово

Входной файл	стандартный ввод / input.txt
Выходной файл	стандартный вывод / output.txt
Ограничение времени (сек/тест)	1
Ограничение памяти (Мб)	64
Максимальный балл	25+10 ^(*)

Условие задачи

Учитель Вовочки по английскому языку пишет кандидатскую диссертацию по филологии на тему, связанную со словообразованием. В качестве примеров к одному из разделов диссертации необходимо привести длинные слова. Учитель решил привлечь к поиску таких слов своих учеников и пообещал поставить пятерку тому, кто назовет ему самое длинное слово. Вовочка в течение последнего месяца интенсивно готовился к олимпиаде по информатике, не выполнял домашние задания по другим предметам, получил несколько двоек и поэтому очень хочет получить пятерку для выправления ситуации с оценками. Искать длинные слова Вовочка планирует в наборе текстов, которые он скачал из Интернета. Вовочка просит вас помочь ему написать программу для поиска длинных слов.

Словом считается любая непрерывная последовательность букв английского алфавита, цифр и символов подчеркивания '_', регистр букв (заглавные или строчные) значения не имеет. Вам необходимо найти самое длинное слово в тексте.

Входные данные

Текст записан во входном файле в виде множества строк. Среди строк могут быть пустые или состоящие только из одних пробелов и знаков табуляции, однако известно, что хотя бы одно слово в тексте обязательно присутствует. В конце текста может не быть перевода на новую строку.

Длина строки может быть произвольной, но известно, что общий объем текста не превышает 1 Мб.

Выходные данные

В первой строке выходного файла необходимо вывести найденное слово максимальной длины в том же регистре букв, с которым данное слово встречается в тексте.

Если в тексте максимальной длине слова соответствуют несколько различных слов, то вывести можно любое из них.

Пример входного файла (stdin / input.txt)	Пример выходного файла (stdout / output.txt)
London is the capital of Great Britain	capital
678+23467.89+87909-234_L/43 // Error	234_L
Voronezh State University Voronezh State University is one of the main universities in Central Russia.	universities

(*) – дополнительные 10 баллов начисляются за абсолютно правильное решение задачи (решение считается абсолютно правильным, если оно проходит все тесты)

Решение жюри

```
#include <fstream>

using namespace std;

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

    string max_word, word;
    while (! cin.eof()) {
        char ch = cin.get();
        if ('a' <= ch && ch <= 'z' ||
            'A' <= ch && ch <= 'Z' ||
            '0' <= ch && ch <= '9' ||
            ch == '_')
            word += ch;
        else {
            if (word.size() > max_word.size())
                max_word = word;
            word = "";
        }
    }
    if (word.size() > max_word.size())
        max_word = word;

    cout << max_word.c_str() << endl;
}
```

Задача 2. Деление с заданной точностью

Входной файл	стандартный ввод / input.txt
Выходной файл	стандартный вывод / output.txt
Ограничение времени (сек/тест)	1
Ограничение памяти (Мб)	64
Максимальный балл	25+10

Условие задачи

Вовочка увлекается астрономией. Для расчета положения далекой кометы, которую он сегодня ночью хочет увидеть в телескоп, ему требуются очень точные вычисления. Точности 8-байтовых вещественных чисел для этой задачи недостаточно. Вовочка просит вас помочь ему реализовать задачу деления двух чисел с заданной точностью.

Даны два натуральных числа A и B ($1 \leq A \leq 1000000$, $1 \leq B \leq 1000000$). Необходимо найти частное A/B этих чисел с точностью N ($0 \leq N \leq 1000$) знаков после запятой. Последний (N -ый после запятой) знак в полученном результате округляется согласно правилам математического округления.

Правые незначащие нули при выводе результата необходимо отбросить.

Входные данные

В первой строке входного файла записаны через пробел три числа – A , B и N .

Выходные данные

В первой строке выходного файла необходимо записать полученный результат, отбросив, если присутствуют, правые незначащие нули. В качестве символа-разделителя целой и дробной части использовать точку.

Примечание

Использовать в реализации данной задачи встроенные средства языка программирования для работы с числами произвольной точности, включая стандартные библиотеки, запрещено (например, класс `BigDecimal` в Java и т.п.).

Пример входного файла (stdin / input.txt)	Пример выходного файла (stdout / output.txt)	Комментарий
5 3 3	1.667	3-ий знак после запятой, цифра 7, получен в результате округления (т.к. 4-ый знак в более точном представлении результата (1.6666666...) – $6 > 5$)
1 2 5	0.5	Незначащие нули при выводе результата (0.50000) отброшены
6 3 1000	2	Незначащие нули при выводе результата также отброшены, включая точку
345 78 25	4.4230769230769230769230769	

Решение жюри

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

using namespace std;

int main(int argc, char *argv[])
{
    ifstream cin("input.txt");
    ofstream cout("output.txt");

    int a, b, n;
    cin >> a >> b >> n;

    stringstream ss;
    ss << a / b << '.';
    a %= b;
    for (int i = 0; i <= n; i++) {
        a *= 10;
        ss << a / b;
        a %= b;
    }

    string s = "0";
    s += ss.str();

    bool o = s[s.size() - 1] != '.' && s[s.size() - 1] >= '5';
    s.resize(s.size() - 1);
    for (int i = s.size() - 1; o && i >= 0; i--) {
        if (s[i] == '.')
            continue;

        o = s[i] == '9';
        s[i] = o ? '0' : s[i] + 1;
    }

    for (int i = s.size() - 1; i > 0 && s[i] == '0'; i--)
        s.resize(s.size() - 1);
    if (s[s.size() - 1] == '.')
        s.resize(s.size() - 1);

    for (; s.size() > 1 && s[0] == '0' && s[1] != '.'; )
        s.erase(0, 1);

    cout << s << endl;
}
```

Задача 3. Волшебный эликсир

Входной файл	стандартный ввод / input.txt
Выходной файл	стандартный вывод / output.txt
Ограничение времени (сек/тест)	1
Ограничение памяти (Мб)	64
Максимальный балл	25+10

Условие задачи

Прошел день, закончился вечер, Вовочка выключил компьютер, улегся в кровать и тут же заснул. Ему приснился сон, в котором он путешествовал по подземелью. Перед ним закрытая дверь, Вовочка пытается ее открыть, у него получается и он оказывается в пещере.

Пещера выглядела бы вполне обычно, если бы не гномы, устроившие в ней потасовку. Невысокие существа с длинными бородами бегали по пещере, толкали друг друга и громко ругались. Разобрав речь гномов, Вовочка понял, что они не поделили волшебный эликсир. У каждого гнома был сосуд с эликсиром, но количество эликсира в этих сосудах было разное. Вовочка не смог пройти мимо и решил помирить гномов, пообещав им, что разделит эликсир между ними поровну. Гномы с таким предложением согласились.

Волшебный эликсир – необычное вещество (на то он и волшебный). При переливании его из одной емкости в другую K процентов переливаемого вещества испаряется. Более формально: если переливать из одного сосуда в другой эликсир в количестве V единиц, то в первом сосуде (из которого переливают) количество вещества уменьшается на V единиц, а во втором сосуде (в который переливают) количество вещества увеличивается на $V * (1 - K / 100)$ единиц.

Так как волшебный эликсир очень тяжело добыть, гномы хотят, чтобы при переливаниях эликсира при его делении испарилось как можно меньше вещества, чтобы у каждого из них в итоге оказалось как можно больше волшебного эликсира.

Помогите Вовочке узнать, какое максимальное возможное одинаковое количество волшебного эликсира он может пообещать гномам после равномерного распределения эликсира между ними.

Входные данные

В первой строке входного файла записаны через пробел два целых числа – N и K ($1 \leq N \leq 10000, 0 \leq K \leq 100$) – количество гномов и процент эликсира, теряющийся при переливании.

В второй строке входного файла записаны N целых чисел A_1, A_2, \dots, A_N – количество волшебного эликсира у каждого гнома ($0 \leq A_i \leq 1000, 1 \leq i \leq N$).

Выходные данные

В единственной строке выходного файла необходимо вывести число C – максимальное одинаковое количество волшебного эликсира, которое можно получить у каждого гнома путем переливания эликсира между сосудами.

Абсолютная погрешность ответа не должна превышать 10^{-6} .

Пример входного файла (stdin / input.txt)	Пример выходного файла (stdout / output.txt)	Комментарий
3 50 8 4 2	4.000000	У 1-го гнома заберем 4 единицы эликсира, а 3-му попадет из них только 2 единицы (50% потерь).
2 90 1 11	1.909091	

Решение жюри

```
#include <fstream>
#include <ios>
#include <iomanip>
#include <vector>
#include <cmath>
#include <algorithm>

using namespace std;

int main()
{
    ifstream cin("input.txt");
    ofstream cout("output.txt");
    cout << fixed << setprecision(10);

    int n;
    double k;
    vector<int> a;

    cin >> n >> k;
    k = (100 - k) / 100;
    a = vector<int>(n);
    for (int i = 0; i < n; i++)
        cin >> a[i];

    double min_bound = a[0], max_bound = a[0];
    for (int i = 0; i < n; i++) {
        min_bound = min(min_bound, (double) a[i]);
        max_bound = max(max_bound, (double) a[i]);
    }

    while (abs(max_bound - min_bound) > 1e-10) {
        double avg = (max_bound + min_bound) / 2,
            diff = 0;
        for (int i = 0; i < n; i++)
            diff += (a[i] - avg) * (a[i] > avg ? k : 1);

        if (diff > 0)
            min_bound = avg;
        else
            max_bound = avg;
    }

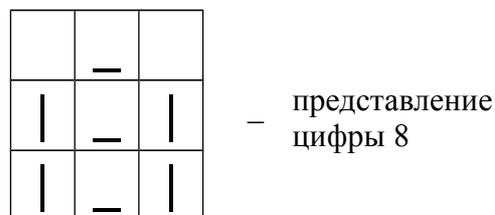
    double result = (max_bound + min_bound) / 2;
    cout << result << endl;
}
```

Задача 4. Сложение чисел ++

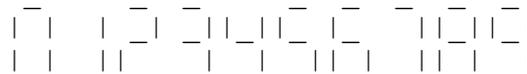
Входной файл	стандартный ввод / input.txt
Выходной файл	стандартный вывод / output.txt
Ограничение времени (сек/тест)	1
Ограничение памяти (Мб)	64
Максимальный балл	25+10

Условие задачи

Как-то раз в первом класса на уроке математики Вовочка, вместо того, чтобы решать примеры, рисовал в тетради в клеточку цифры. Все нарисованные цифры выглядели как на дисплее калькулятора (см. рисунок), т.е. одна цифра занимала ровно 9 клеточек. Цифры располагались в несколько рядов (строк).



Учитель заметил, что Вовочка занимается ерундой, и в качестве наказания (иначе обещал поставить двойку) предложил Вовочке сосчитать сумму чисел, которые образовывали цифры в каждом ряду. Помогите Вовочке справиться с этим заданием.



Входные данные

Входной файл состоит из нескольких строк одинаковой длины (в символах) – для записи расположения чисел, получившегося у Вовочки. При этом вертикальные отрезки нарисованных цифр записываются символом вертикальной черты – '|', а горизонтальные – символом подчеркивания; пустое место обозначается пробелом. Любая цифра состоит из 9 символов, на рисунке показана запись цифры 8 в таком формате. Остальные цифры записываются аналогично – см. рисунок. Максимальное количество цифр в каждом числе – 9. Между цифрами могут быть произвольные расстояния, которые описываются одинаковым количеством пробелов во всех 3-х строках конкретного ряда цифр. При этом длина строк вместе с пробелами, включая конечные, – не более 100 символов каждая. Между рядами цифр также может быть произвольное число строк, состоящих из одних пробелов. Общее количество строк, включая состоящие из одних пробелов, также не более 100.

Выходные данные

В выходной файл требуется вывести единственное число – сумму чисел, составленных из цифр каждого ряда. Известно, что полученная сумма не более 10^9 .

Пример входного файла (stdin / input.txt)	Пример выходного файла (stdout / output.txt)
<pre> _ _ _ _ _ _ _ _ _ _ _ _ </pre>	<p>1511</p>

Решение жюри

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

using namespace std;

const string digits =
    (string) "          " +
    (string) "|_| |_| |_|_|_|_|_|_|_|_|_|_|" +
    (string) "|_| |_| |_|_|_|_|_|_|_|_|_|_|" ;

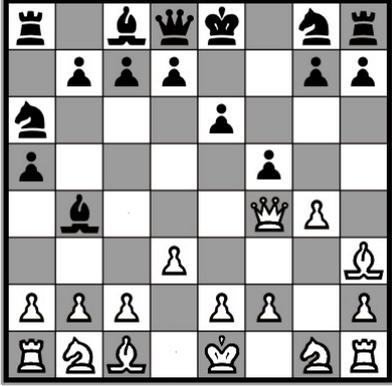
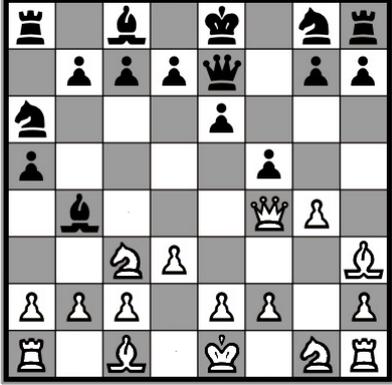
int main(int argc, char *argv[])
{
    ifstream cin("input.txt");
    ofstream cout("output.txt");

    vector<string> data;

    while (!cin.eof()) {
        string str;
        getline(cin, str);
        data.push_back(str);
    }

    int sum = 0;
    for (int r = 0; r < data.size() - 2; r++) {
        string num;
        for (int c = 0; c < data[r].size() - 2; c++) {
            for (int i = 0; i < 10; i++) {
                bool found = true;
                for (int rr = 0; rr < 3; rr++)
                    for (int cc = 0; cc < 3; cc++)
                        if (data[r + rr][c + cc] !=
                            digits[rr * digits.size() / 3 + 3 * i + cc])
                            found = false;
                if (found && i == 1 &&
                    c < data[r].size() - 3 && data[r][c + 3] != ' ')
                    found = false;
                if (found)
                    num += '0' + i;
            }
        }
        if (num.size())
            sum += atoi(num.c_str());
    }

    cout << sum << endl;
}
```


Пример входного файла (stdin / input.txt)	Пример выходного файла (stdout / output.txt)	Комментарий
<pre>r.bqk.nr .ppp.pp n...p... p....p.. .b...QP. ...P...B PPP.PP.P RNB.K.NR</pre>	<pre>2 0 0 0 0 0 0 1 0 0 0 0 2 1 2 0 0</pre>	 <p>Белому королю <i>объявлен шах</i> (черным слоном), поэтому очередной ход белых должен данный шах устранить.</p> <p>Для этого необходимо или <i>уйти от шаха</i> (сделать ход королем на одну из соседних пустых белых клеток) или <i>закрыться от шаха</i>, т.е. от черного слона (пешкой – 1 ход, конем – 2 возможных хода, слоном – 1 ход, ферзем – 1 ход) или <i>взять</i> слона ферзем (т.е. всего ферзем можно сделать 2 различных хода).</p>
<pre>r.b.k.nr .pppq.pp n...p... p....p.. .b...QP. ..NP...B PPP.PP.P R.B.K.NR</pre>	<pre>14 2 0 1 2 2 1 0 2 1 0 1 2 3 1 0</pre>	 <p>В данной ситуации от <i>шаха</i> белого короля <i>прикрывает</i> конь (рядом с черным слоном), следовательно, этим конем сделать ход нельзя.</p> <p>Остальными белыми фигурами можно выполнить любой допустимый ход в соответствии с правилами для данной фигуры (некоторые белые фигуры «зажаты» и не могут сделать ни одного хода).</p>

Решение жюри

```
#include <fstream>

using namespace std;

const int CB_SIZE = 8;

typedef struct {
    char f; // figure
    bool w; // white
} chessboard_item_t;

typedef chessboard_item_t chessboard_t[CB_SIZE][CB_SIZE];
typedef bool positions_t[CB_SIZE][CB_SIZE];

typedef struct {
    int dr, dc;
} direction_t;

direction_t d[8] = { { 1, 0 }, { 0, 1 }, { -1, 0 }, { 0, -1 },
                    { 1, 1 }, { -1, 1 }, { -1, -1 }, { 1, -1 } };

template <typename T> inline bool between(T v, T a, T b) {
    return a <= v && v <= b;
}

inline bool is_pos_correct(int row, int col) {
    return between(row, 0, CB_SIZE - 1) && between(col, 0, CB_SIZE - 1);
}

void clear_positions(positions_t p) {
    for (int r = 0; r < CB_SIZE; r++)
        for (int c = 0; c < CB_SIZE; c++)
            p[r][c] = false;
}

void get_positions_for_figure(chessboard_t cb, int row, int col, positions_t p)
{
    if (cb[row][col].f == '.')
        return;

    switch (cb[row][col].f) {
        case 'K': case 'k': // king
        case 'Q': case 'q': // queen
        case 'R': case 'r': // rook
        case 'B': case 'b': // bishop
        {
            int from, to, count;

            switch (cb[row][col].f) {
                case 'K': case 'k': // king
                    from = 0; to = 7; count = 1; break;
                case 'Q': case 'q': // queen
                    from = 0; to = 7; count = 7; break;
                case 'R': case 'r': // rook
                    from = 0; to = 3; count = 7; break;
                case 'B': case 'b': // bishop
                    from = 4; to = 7; count = 7; break;
            }

            for (int i = from; i < to + 1; i++) {
```

```
for (int j = 1; j < count + 1; j++) {
    int r = row + j * d[i].dr,
        c = col + j * d[i].dc;

    if (!is_pos_correct(r, c))
        break;

    if (cb[r][c].f == '.' || cb[r][c].w != cb[row][col].w) {
        p[r][c] = true;
        if (cb[r][c].f != '.')
            break;
    }
    else
        break;
}

break;
}
case 'N': case 'n': // knight
{
    for (int i = 4; i < 8; i++)
        for (int j = 1; j < 3; j++) {
            int r = row + j * d[i].dr,
                c = col + (3 - j) * d[i].dc;

            if (is_pos_correct(r, c))
                if (cb[r][c].f == '.' || cb[r][c].w != cb[row][col].w)
                    p[r][c] = true;
        }

    break;
}
case 'P': case 'p': // pawn
{
    int dr = cb[row][col].w ? -1 : 1,
        count = cb[row][col].w && row == CB_SIZE - 2 ||
                !cb[row][col].w && row == 1
                ? 2 : 1;

    for (int i = 1; i < count + 1; i++) {
        int r = row + i * dr,
            c = col;

        if (!is_pos_correct(r, c))
            break;

        if (cb[r][c].f == '.')
            p[r][c] = true;
        else
            break;
    }

    for (int dc = -1; dc < 2; dc++)
        if (dc != 0) {
            int r = row + dr,
                c = col + dc;

            if (is_pos_correct(r, c))
                if (cb[r][c].f != '.' && cb[r][c].w != cb[row][col].w)
                    p[r][c] = true;
        }

    break;
}
}
}
```

```
void get_all_positions(chessboard_t cb, bool white, positions_t p) {
    clear_positions(p);

    for (int r = 0; r < CB_SIZE; r++)
        for (int c = 0; c < CB_SIZE; c++)
            if (cb[r][c].f != '.' && cb[r][c].w == white)
                get_positions_for_figure(cb, r, c, p);
}

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

    chessboard_t cb;
    int wk_row = -1, wk_col = -1;

    for (int r = 0; r < CB_SIZE; r++)
        for (int c = 0; c < CB_SIZE; c++) {
            cin >> cb[r][c].f;
            cb[r][c].w = between(cb[r][c].f, 'A', 'Z');

            if (cb[r][c].f == 'K') {
                wk_row = r;
                wk_col = c;
            }
        }

    for (int r = 0; r < CB_SIZE; r++)
        for (int c = 0; c < CB_SIZE; c++)
            if (cb[r][c].f != '.' && cb[r][c].w) {
                int steps_count = 0;
                positions_t p;
                clear_positions(p);

                get_positions_for_figure(cb, r, c, p);
                for (int r2 = 0; r2 < CB_SIZE; r2++)
                    for (int c2 = 0; c2 < CB_SIZE; c2++)
                        if (p[r2][c2])
                            if (is_pos_correct(wk_row, wk_col))
                                {
                                    chessboard_item_t cbi = cb[r2][c2];
                                    cb[r2][c2] = cb[r][c];
                                    cb[r][c].f = '.';

                                    positions_t p2;
                                    get_all_positions(cb, false, p2);
                                    if ((r != wk_row || c != wk_col) && !p2[wk_row][wk_col] ||
                                        r == wk_row && c == wk_col && !p2[r2][c2])
                                        steps_count++;

                                    cb[r][c] = cb[r2][c2];
                                    cb[r2][c2] = cbi;
                                }
                            else
                                steps_count++;

                cout << steps_count << endl;
            }
        }
}
```