Региональная предметно-методическая комиссия по информатике
Тула

Ключи к заданиям муниципального этапа всероссийской олимпиады школьников 2024/2025 учебного года по информатике

Задача 1. Реликвии бобр-паладинов

Технические требования:

Ограничение времени	2 секунды на один тест
Ограничение памяти	100.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Описание

Гильдия искателей Бобргарда нашли магические реликвии, что способны многократно усилить могущество бобр-паладинов. Применение нескольких реликвий перемножает их эффект между собой, так реликвия с силой 2 увеличивает могущество паладина в 2 раза, а использование двух реликвий с силами 2 и 5 увеличивает могущество паладина в 10 раз. Однако некоторые реликвии вместо увеличения сил паладина множат входящий урон по броне паладина. Использование ранее созданного положительного эффекта с отрицательным эффектом даст отрицательный эффект. В то же время применение двух реликвий с отрицательным эффектом дает положительный эффект как от таких же реликвий, но с положительным эффектом. Одновременно бобр-паладин может задействовать лишь три реликвии. Подберите из всего доступного множества реликвий лучший набор для бобр-паладина.

Входные данные

Вводится сначала число N - количество реликвий доступных для выбора ($3 \le N \le 100$). Далее записана сама последовательность сил реликвий: N целых чисел, по модулю не превышающих 1000.

Выходные данные

Выведите три искомых силы реликвий в любом порядке. Если существует несколько различных наборов из трех реликвий, дающих максимальный эффект, то выведите любой из них.

Примеры файлов входных и выходных данных:

Ввод	*Вывод*
9	9 9 10

3 5 1 7 9 0 9 -3 10	

Решение

```
#include <iostream>
#include <vector>
#include <algorithm>

int main() {
    size_t n;
    std::cin >> n;
    std::vector<short> v;
    v.reserve(n);
    for (size_t i = 0; i < n; ++i) {
        short item;
        std::cin >> item;
        v.emplace_back(item);
    }
    std::sort(v.begin(), v.end());
    const auto p1 = (long long)v[0] * v[1] * v[n - 1];
    const auto p2 = (long long)v[n - 3] * v[n - 2] * v[n - 1];
    if (p1 > p2) {
        std::cout << v[0] << ' ' << v[1] << ' ' << v[n - 1] <<
'\n';
    } else {
        std::cout << v[n - 3] << ' ' << v[n - 2] << ' ' << v[n - 1]
<< '\n';
    }
    return 0;
}
```

Задача 2. Ленивая белка

Технические требования:

Ограничение времени	2 секунды на один тест
Ограничение памяти	100.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Описание

В хвойном лесу для некоторых деревьев верно, что с ветки одного дерева белка может перепрыгнуть на ветку другого дерева, при этом ветки таких деревьев находятся на одном уровне (белка может прыгнуть обратно), а мы говорим, что эти деревья связаны. Ленивая белка хочет расположить свои убежища таким образом, чтобы собирать орехи с деревьев в одном убежище не опускаясь на землю, потому что она ленится подниматься на дерево и спускаться с него с орехами.

Помогите ленивой белке посчитать сколько ей придется делать убежищ.

Входные данные

Введены сначала два числа N и M, задающие соответственно количество деревьев и количество ветвей соединяющих деревья ($1 \le N \le 100$, $0 \le M \le 10000$), а затем перечисляются ветви. Каждая ветвь задается номерами деревьев, которые она соединяет.

Выходные данные

Выведите одно число - количество убежищ ленивой белки.

Примеры файлов входных и выходных данных:

Ввод	*Вывод*
3 4 1 1 1 2 1 3 2 3	1
5 3 1 1 1 2 2 1	4
5 0	5

Решение

Подойдет любой алгоритм поиска компонент связностей графа.
DSU подобная структура(ленивый подход)

Будем делать конденсацию компоненты в одну вершину. Идея следующая: будем последовательно обрабатывать ребра. Каждая компонента связности будет представлена одной своей вершиной(титульная). При этом неважно какой. По ходу обработки ребер титульная вершина компонент может меняться. В итоге для нахождения количества компонент связности нужно найти количество титульных вершин.

Оптимизация

При добавлении нового ребра будем “мерджить меньшее подмножество к большему” + сжимать пути.

```
#include <iostream>
#include <bits/stdc++.h>
int n,m,f,s;
const int SIZE = 1e3 + 10;

int p[SIZE];
int size[SIZE];

int par(int x) {
  return p[x] == x ? x : p[x] = par(p[x]);
}

int connected_components_amount_dsu() {

  scanf("%d %d", &n, &m);

  for (int i=1;i<=n;++i) {
    p[i] = i;
    size[i] = 1;
  }

  for (int i=0;i<m;++i) {
    scanf("%d %d", &f, &s);
    f = par(f); s = par(s);
    if (f != s) {
      if (size[f] > size[s])
        std::swap(f,s);
      p[f] = s;
      size[s] += size[f];
    }
  }
  bool usd[SIZE];
```

```
  memset(usd, 0, sizeof(usd));
  for (int i=1;i<=n;++i)
    usd[par(i)] = true;
  int cnt = 0;
  for (int i=1;i<=n;++i) {
    if (usd[i]) ++cnt;
  }

  return cnt;
}
int main()
{
    printf("%d",connected_components_amount_dsu());
    return 0;
}
```

Задача 3. Привередливая белка

Технические требования:

Ограничение времени	1 секунды на один тест
Ограничение памяти	20.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Описание

В хвойном лесу растут ели, сосны и пихты. Ветви деревьев соприкасаются между собой. Привередливой беке нравится перепрыгивать с веток дерева на ветки другого дерева той же породы (с ели на другую ель, но не на пихту или сосну). Деревья считаются связанными если с ветки первого дерева можно перепрыгнуть на ветки второго дерева, а место где белка может перепрыгнуть с первого дерева на второе дерево будем называть переходом. Требуется определить сколько "неправильных" переходов с точки зрения привередливой белки в лесу есть в лесу.

Входные данные

В первой строке записано N ($0 < N \leq 1000$) - число деревьев. Далее идет матрица смежности, описывающая наличие переходов между деревьями (1 - переход есть, 0 - нет).

В последней строке записано N чисел, обозначающих породу деревьев:

1 - ель;

2 - сосна;

3 - пихта.

Выходные данные

вести количество "неправильных" переходов.

Примеры файлов входных и выходных данных:

Ввод	*Вывод*
8 0 1 0 0 1 0 1 1 0 0 0 1 0 1 1 1 1 0 0 0 1 0 1 1 0 1 1 1 0 1 1 0 1 0 0 1 0 1 0 0 0 0 1 0 1 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 0 0 0 1 1 2 1 1 2 1 2 3	12

Решение

```
program CountInvalidTransitions;

const
    MaxN = 100;
    INF = 1000000000; // "бесконечность"

type
    Matrix = array[1..MaxN, 1..MaxN] of longint; // Тип матрицы
смежности. M[i, j] = true, если существует ребро, идущее от вершины
i к j
    Color = array[1..MaxN] of longint;           // Массив
цветов (вершин)

var
    Hills: Matrix;
    N: integer;
    ColorHill: Color;

procedure Input_Table(var Hills: Matrix; N: longint); //
Процедура ввода матрицы смежности A(N, N) из консоли
var
    i, j: longint;
begin
    for i := 1 to N do
    begin
        for j := 1 to N do
        begin
```

```
                read(Hills[i, j]);
                if (Hills[i, j] = 0) and (i <> j) then Hills[i, j]
:= INF; // Вершины, которые не связаны ребром, будем обозначать
"бесконечностью" ввиду ограничения на вес рёбер
            end;
            readln;
        end;
        for i := 1 to N do
        begin
            read(ColorHill[i]);
            //write(ColorHill[i], ' ');
        end;
        //writeln;
    end;

    procedure Plohie_mosti(var Hills: Matrix; N: longint); //
Процедура для подсчёта неправильных переходов
    var
        i, j: longint;
        m, k: integer;
    begin
        k := 0;
        for i := 1 to N do
        begin
            m := Hills[i, i];
            for j := 1 to N do
            begin
                if (Hills[i, j] = 1) and (ColorHill[i] <>
ColorHill[j]) then k := k + 1
            end;
        end;
        k := k div 2;
        writeln(k);
    end;

    begin
        readln(N);
        Input_Table(Hills, N);
        Plohie_mosti(Hills, N);
end.
```

Задача 4. Щепки летят

Технические требования:

Ограничение времени	1 секунды на один тест
Ограничение памяти	20.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Описание

Бобры решили показать белкам, кто в лесу хозяин. Для этого они выбрали квадратный участок леса, на котором деревья росли строго как на поле в клетку по одному дереву в каждой клетке (бывает же такое!). Бобры расположились на этом участке, каждый около своего дерева, которое было моментально срублено. Далее по команде Бобра-прораба они начали рубку леса. За один час каждый бобёр рубит все соседние рядом с ним по горизонтали и вертикали деревья. Например, на рисунке показано, сколько часов понадобится бобру, находящемуся в центре участка размером 5x5, чтобы срубить каждое дерево.

4	3	2	3	4
3	2	1	2	3
2	1	бобёр	1	2
3	2	1	2	3
4	3	2	3	4

Требуется написать программу, которая поможет Бобру-прорабу определить минимальное время, необходимое M бобрам для вырубки леса на участке размером N x N деревьев.

Входные данные

В первой строке содержатся два натуральных числа N ($1 \le N \le 500$) и M ($1 \le M \le 10$), задающие размер участка леса (т.е. количество деревьев по горизонтали и по вертикали) и количество работающих бобров.

Далее в M строках содержится по два целых неотрицательных числа X ($0 \le X \le 499$) и Y ($0 \le Y \le 499$), задающих координаты каждого работающего бобра (координаты отсчитываются от нуля и по горизонтали, и по вертикали).

Выходные данные

Вывод содержит единственное целое неотрицательное число, равное минимальному количеству часов, необходимых работающим бобрам для вырубки всего участка леса.

Примеры файлов входных и выходных данных:

Ввод	*Вывод*
5 1 2 2	4
10 3 0 0 4 5 9 7	8

Решение

Дерево с координатами (i,j) будет срублено бобром, находящимся на позиции (X,Y) за количество часов H, которое нетрудно рассчитать по формуле:

H=|i-X|+|j-Y|.

Данная формула позволяет определить, через какое время конкретное дерево будет срублено конкретным бобром. Однако, бобров на участке леса может быть несколько. Поэтому определяем время рубки каждым бобром, а находим минимальное из них, которое и сохраняем в массиве.

Далее необходимо найти максимальное значение среди вычисленных и сохраненных минимальных значений вырубки для каждого дерева.

```
#include <iostream>
#include <math.h>

using namespace std;

#define N 500
#define M 10

typedef struct bob
{
  int x;
  int y;
};

int main()
{
  int p[N][N];
  bob b[M];
  int tr, bobr, x, y, i, j, k, min, t, max;

  cin >> tr;
  cin >> bobr;
```

```
  for (i = 0; i < bobr; i++)
  {
    cin >> b[i].x;
    cin >> b[i].y;
  }

  for (i = 0; i < tr; i++)
    for (j = 0; j < tr; j++)
    {
       min = abs(i - b[0].x) + abs(j - b[0].y);
       for (k = 1; k < bobr; k++)
       {
          t = abs(i - b[k].x) + abs(j - b[k].y);
          if (t < min) min = t;
        }
        p[i][j] = min;
    }

  max = p[0][0];
  for (i = 0; i < tr; i++)
    for (j = 0; j < tr; j++)
      if (p[i][j] > max) max = p[i][j];

  cout << max << endl;
  return 0;
}
```

Задача 5. Из пустого в порожнее

Технические требования:

Ограничение времени	1 секунды на один тест
Ограничение памяти	64.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Описание

Чтобы утереть нос бобрам, белки решили заняться наукой. В рамках научных работ они решают задачу следующего содержания: имеется N кувшинов с целочисленными объемами V_1, ..., V_n литров, пустой сосуд и кран с водой. Можно ли с помощью этих банок налить в сосуд ровно V литров воды?

Требуется написать программу, которая поможет белкам определить, возможно ли при наличии N кувшинов с целочисленными объемами V_1, ..., V_n литров и неограниченного объема доступной воды получить в пустом сосуде ровно V литров воды.

Формат входных данных:

В первой строке содержится натуральное число N ($2 \le N \le 100$), задающее количество кувшинов, и натуральное число V ($1 \le V \le 20000$), определяющее объем воды, который нужно получить в пустом сосуде. В следующей строке содержатся разделенные пробелами N натуральных чисел V_i ($1 \le i \le N$, $1 \le V_i \le 20000$), задающих объем для каждого кувшина.

Формат выходных данных:

Вывод должен содержать одно из чисел 1 или 0 в зависимости от того, можно ли получить требуемый объем в пустом сосуде, или нет.

Пример файлов входных и выходных данных:

Ввод	*Вывод*
2 4 5 3	1

Решение

При помощи двух кувшинов можно получить в сосуде количество воды, равное наибольшему общему делителю их объемов. Например, с помощью кувшинов в 7 и 9 литров, можно получить любое количество воды, так как наибольший делитель (НОД) этих чисел равен единице.

Таким образом, необходимо вычислить НОД всех чисел согласно формуле:

НОД(A, B, C) = НОД (НОД(A, B), C)

Для нахождения НОД опишем рекурсивную функцию, которая возвращает первое число в качестве результата, если второе равно нулю, в противном случае вычисляем остаток от деления первого числа на второе и заменяем им второе, а первое число заменяем вторым.

Если требуемый объем кратен найденному наибольшему общему делителю, то значит, мы сможем получить его. В этом случае выводим 1, в противном случае выводим 0.

```
Program belka1;
  function nod(a,b:integer):integer;
  begin
    if b<=0 then nod:=a
    else nod:=nod(b, a mod b);
  end;
  var f:text; N,V,v0,v1,i,nd:integer;
Begin
  read(N);
  read(V);
  read(v0);
  read(v1);
  nd:= nod(v0,v1);
  for i:=2 to N do
  begin
    read(v1);
    nd:= nod(nd,v1);
  end;
  if V mod nd = 0 then writeln('1')
  else writeln('0');
  readln;
End.
```

Задача 6. Думаем о будущем

Технические требования:

Ограничение времени	1 секунды на один тест
Ограничение памяти	64.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Описание

На каждом занятии в лесной школе Бобёр Радомир и Белочка Дарина становились все умнее и умнее. Их любимым предметом была математика. И со временем они так вошли во вкус, что начали видеть вокруг себя полезные математические модели. Сначала они осознали, что их родной лес представляет собой прямоугольник, составленный из единичных квадратов. А потом поняли, что квадраты похожи друг на друга только своим внешним видом и площадью, но отличаются запасами всяких разных полезных штучек: деревья, грибы, ягоды, орешки, вкусные травки и т.д. и неудобств в виде буреломов, чащ, болот и оврагов. Радомир и Дарина решили провести аудит этих запасов в балльной системе. Они оценили все достоинства каждого квадрата леса натуральным числом, а размер неудобств – отрицательным и выставили общую оценку.

Узнал об этом Хозяин леса и решил навести порядок в учете и использовании запасов и наградить Бобра и Белочку, проверив их при этом на сообразительность.

Хозяин дал им возможность выбора для проживания одного из нескольких предложенных прямоугольных участков леса и уточнил, что подтвердит этот выбор только в том случае, если Радомир и Дорина смогут правильно определить прямоугольник с наибольшим количеством баллов.

Если будет несколько таких прямоугольников, то необходимо указать тот, который в списке предложений **был назван первым**.

Эта задача оказалась для Радомира и Дорины не очень простой, и они решили изучить программирование для того, чтобы суметь воспользоваться современными информационными технологиями и получить желаемую награду.

Требуется написать программу, которая выполняет поиск прямоугольной области леса, имеющей наибольшую сумму баллов.

Формат входных данных:

Ввод содержит следующие последовательности строк. В первой строке записаны числа N и M ($1 <= N <= 100$, $1 <= M <= 100$) – число квадратов вдоль по каждой из двух соседних сторон леса.

Затем в N строках описывается таблица баллов для каждого квадрата леса (в каждой строке M целых чисел, по модулю не превышающих 1000).

В последующих строках содержатся описания предлагаемых к рассмотрению прямоугольников, в которых требуется найти максимальную сумму баллов.

При этом: сначала идет строка с числом таких прямоугольников – K ($1 <= K <= 500000$), после этого идут K строк, задающих сами прямоугольники. В

каждой строке последовательно записаны четыре числа: номер столбца верхней клетки прямоугольника, номер строки верхней клетки прямоугольника, номер столбца нижней клетки прямоугольника, номер строки нижней клетки прямоугольника. Столбцы нумеруются, начиная с единицы, слева направо, строки – сверху вниз. Числа в каждой строке файла должны быть разделены пробелами.

Формат выходных данных:

Вывод должен содержать два числа в одной строке разделенных пробелом. Первое – вычисленное наибольшее количество баллов, второе – порядковый номер выбранного прямоугольника из предложенного списка.

Пример файлов входных и выходных данных:

Ввод	*Вывод*
3 4 1 2 3 4 5 6 7 8 1 1 1 1 3 1 1 1 1 2 1 3 2 1 1 4 3	40 3
5 7 -1 5 4 -3 0 6 6 5 8 -1 5 8 -4 1 5 4 4 5 6 4 5 -4 6 6 0 1 2 -4 6 4 -2 9 0 -2 4 12 2 2 4 3 1 2 3 2 3 1 4 2 1 1 1 2 1 1 1 1 1 2 1 3 3 2 5 2 1 2 3 2 3 1 3 1 3 1 5 2 2 2 4 2 1 2 1 3	25 1

Решение

Задачу можно решить «в лоб», т.е. после формирования исходной таблицы при помощи считывания данных из файла, последовательно считывать данные, характеризующие координаты вершин нужных нам прямоугольников, а после этого обычными циклическими действиями вычислять сумму чисел в каждом прямоугольнике и обрабатывать данные с целью поиска максимальной суммы. Однако, несложно увидеть, что при этом очень часто будут производиться непродуктивные действия, которые заключаются в том, чтобы суммировать числа в одной и той же области. Это происходит при наложении прямоугольников. Очевидно, что время работы программы выходит за пределы ограничений.

Для того, чтобы избежать непродуктивных действий, предлагаем рассмотреть следующую модель для решения. Имеется вспомогательный массив **dop_mass**, элементы которого используются для хранения суммы элементов исходного **mas_date** массива, расположенных не ниже **i** – строки и не правее **j** – столбца.

	0		**j-1**	**j**		**m**
0						
i-1				**dop_mass[i-1,j]**		
i			**dop_mass[i,j-1]**	**dop_mass[i,j]**		
n						

Индексы строки и столбца в данном массиве будут изменяться от нуля. Нулевая строка и нулевой столбец будут заполнены нулями. Это необходимо для того, чтобы унифицировать заполнение значений элементов первой строки и левого столбца по формуле:

dop_mas[i,j]:=dop_mas[i-1,j]+dop_mas[i,j-1]-dop_mas[i-1,j-1]+ mas_date [i,j].

Очевидно, что для вычисления значения dop_mas[i,j] надо сложить значения **dop_mass[i-1,j]** и **dop_mass[i,j-1].** При этом важно обратить внимание на тот факт, что элемент, а значит и соответствующая сумма, **dop_mass[i-1,j-1]** входит в каждое из этих слагаемых. Абсолютно новым приобретением является элемент исходного массива **mas_date [i,j].**

После проведенных вычислений сумму элементов, находящихся внутри каждого прямоугольника определенного числами **L, U, R, D** можно находить при помощи трех действий:

sum:=dop_mas[D,R] – dop_mas[U-1,R] – dop_mas[D,L-1]+dop_mas[U-1,L-1];

		L	**…**	**R**		
	dop_mass[U-1,L-1]			**dop_mass[U-1,R]**		
U			**…**			
…		**…**	**…**	**…**		
D	**dop_mass[D,L-1]**		**…**	**dop_mass[D,R]**		

Контрольные тесты

INPUT.TXT	*OUTPUT.TXT*
4 5 17 8 2 13 7 5 4 7 0 12 1 3 7 12 1 8 4 1 7 2 5 3 1 4 2 1 1 3 4 2 1 2 4 3 1 5 4 1 2 5 4	74 5

```
#include <iostream>
#include <fstream>
#define N 100
using namespace std;

void massiv_is_file(int m[N][N], int a, int b)
{
    for (int i = 0; i < a; ++i)
    {
        for (int j = 0; j < b; ++j)
        {
            cin >> m[i][j];
        }
    }
}

void dop_masss(int DM[N][N], int M[N][N], int n, int m)
{
```

```
    for (int i = 0; i <= n + 1; i++)
        for (int j = 0; j <= m + 1; j++)
        {
            if ((i == 0) || (j == 0))
                DM[i][j] = 0;
            else
                DM[i][j] = DM[i - 1][j] + DM[i][j - 1] - DM[i - 1][j - 1] + M[i - 1][j - 1];
        }
}

int main() {
    int n, m, k, nomer = 1;
    int mas_date[N][N];
    int dop_mass[N][N];
    cin >> n >> m;
    massiv_is_file(mas_date, n, m);
    dop_masss(dop_mass, mas_date, n, m);
    //cout << endl;
    int L, U, R, D;
    cin >> k;
    cin >> L >> U >> R >> D;
    int maximum;
    maximum = dop_mass[D][R] - dop_mass[U - 1][R] - dop_mass[D][L - 1] + dop_mass[U - 1][L - 1];
    for (int i = 2; i <= k; i++)
    {
        cin >> L >> U >> R >> D;
```

```
        int summa = dop_mass[D][R] - dop_mass[U - 1][R] -
dop_mass[D][L - 1] + dop_mass[U - 1][L - 1];
        if (summa > maximum) {
            maximum = summa;  nomer = i;
        }
    }
    cout << maximum << " " << nomer;

    return 0;
}
```

Задача 7. Пропуск на игру

Технические требования:

Ограничение времени	1 секунды на один тест
Ограничение памяти	64.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Описание

Бобёр Радомир и Белочка Дарина стали в лесной школе страны Шишколандии самыми прилежными учениками, и учитель мудрости Сова часто поручала им организацию веселых состязаний с другими школьниками. Любимым их развлечением стали математические головоломки. Для того, чтобы получить пропуск для игры на лесной полянке, необходимо отыскать ключевое слово путем решения задания.

В стране Шишколандия алфавит состоит из 10 букв, при этом порядковые номера символов изменяются от 0 до 9. Для испытания поочередно берутся числа, количество разрядов в записи которых изменяется от 1 до 255. Радомир складывает все цифры в каждом числе и получает новые числа. Затем он складывает все цифры в каждом из вновь полученных чисел и продолжает процесс до тех пор, пока в результате не останется число, меньшее 10. Затем Дарина по порядку заменяет полученные числа на соответствующую букву алфавита.

Требуется написать программу, которая будет отыскивать ключевое слово.

Формат входных данных:

Ввод содержит следующие последовательности строк.

В первой строке записано слово – алфавит страны Шишколандии: десять букв расположенных по возрастанию порядковых номеров без пробелов.

Во второй строке записано число *N* (*N* <= 255).

Затем в *N* строках записаны собственно исходные числа (по одному на строке, в каждом не более 255 цифр).

Формат выходных данных:

Вывод должен содержать только одно ключевое слово

Пример файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
KLMOAGEIRT 4 910000 84724 333333333 72517346	LITR
KLMOAGEIRT 6 0000000000 8282828282824 1111111 7733 7984714 636327	KLIMAT

Решение

Читаем в переменную string алфавит государства и количество чисел для поиска ключевого слова.

Затем поочередно считываем в строку из файла числа и для каждого числа (представлено в виде строки) вычисляем сумму его цифр. Потом для полученного числа повторяем процесс вычисления суммы до тех пор, пока она не станет меньше, чем 10. Когда получится число, меньшее 10 - то выведем соответствующую ей букву в output. При выводе символа надо помнить, что символы в строке алфавита занумерованы 0..9.

```
#include <iostream>
#include <fstream>
using namespace std;

int summa_zifr(string chislo)
{
    int temp = 0, i;
     for  (i=0; i<chislo.size(); i++)
     {  temp += chislo[i] - 48;
     }
     return temp;
}

int summa_zifr(int chislo)
{
     int temp = 0;
     while (chislo>0)
     { temp += chislo%10;
     chislo /= 10;
     }
```

```
    return temp;
}


int main()
{
    string alfavit, chislo;
    int kolichestvo, index;
    cin >> alfavit;
    cin >> kolichestvo;
    for (int i=0; i<kolichestvo; i++)
    {
      cin >> chislo;
      index = summa_zifr(chislo);
      while (index>9)
         index = summa_zifr(index);
      cout << alfavit[index];
    }

    return 0;
}
```