El vuelo de Juan

Desde pequeño, el sueño de Juan siempre había sido volar, llegar más allá del cielo, alcanzar las estrellas. Debido a las limitaciones físicas de su condición humana, decide ponerse manos a la obra para construir un coche volador que consiga que sus sueños se hagan realidad.

Tras mucho tiempo estudiando la ingeniería necesaria, y después de muchos intentos fallidos, Juan casi ha logrado construir el vehículo: solo le falta el motor, el volante y la rueda de repuesto (tampoco tiene frenos, pero su optimismo le dice que no los va a necesitar). Muchas tiendas alrededor tienen las tres piezas que necesita, pero por conveniencia decide que solo va a ir a una tienda, la que le salga más barata en total.

¿Podéis ayudarle a decidir a qué tienda debe ir?

Input Format
La entrada empieza con un entero *T*, el número de casos de prueba. Cada caso empieza con *n*, el número de tiendas. Siguen *n* líneas, con 3 enteros *ai, bi, ci* cada una, indicando los precios del motor, el volante y la rueda de repuesto respectivamente en la tienda *i*-ésima.

Constraints
1 <= n <= 10^6, 1 <= ai,bi,ci <= 10^8
Se garantiza que la suma de las *n* en todos los casos no excede *10^7*.

Output Format
Para cada caso, imprimid una línea con el número de la tienda a la que debe ir Juan. Las tiendas se numeran del *1* al *n* según el orden de aparición en la entrada. En caso de empate, elegid la tienda con el número más bajo.

Solución en Python

```
ncases = int(input())

for case in range(ncases):
	basura = input()

	minimo = -1
	mindex = -1
	n = int(input())
	for i in range(n):
		s = input().split(' ')
		s[0], s[1], s[2] = int(s[0]), int(s[1]), int(s[2])

	suma = s[0] + s[1] + s[2]
	if (minimo == -1 or suma < minimo):
		minimo = suma
		mindex = i
	print(mindex+1)
```

Autor de la solución: Javier López-Contreras (Miembro del comité organizador)

Solución en C++

```cpp
#include <iostream>

using namespace std;

int main(){
  int cas;
  cin >> cas;
  while (cas--) {
    int n;
    cin >> n;
    int ind = 0;
    int val = 1e9;
    for (int i = 1; i <= n; ++i) {
      int res = 0;
      for (int j = 0; j < 3; ++j) {
        int x;
        cin >> x;
        res += x;
      }
      if (res < val) {
        val = res;
        ind = i;
      }
    }
    cout << ind << endl;
  }
}
```

Autor de la solución: Martí Oller (Miembro del comité organizador)

Solución en Java

```java
import java.io.*;
import java.util.*;

public class Solution {

	public static void main(String[] args) {
		/* Enter your code here. Read input from STDIN. Print output to STDOUT.
		Your class should be named Solution. */
		int repeticiones = 0;
		int tiendas = 0;
		int barato = 1000000000;
		String precios = "";
		String mejorOpcion = "";
		Scanner n3 = new Scanner(System.in);
		repeticiones = n3.nextInt();
		String aresultados[] = new String [repeticiones];
```

```
                for (int i = 0; i < repeticiones; i++) {
                        tiendas = n3.nextInt();
                        n3.nextLine();
                        barato = 1000000000;
                        for (int j = 0; j <tiendas; j++) {
                                precios = n3.nextLine();
                                String aprecios[] = precios.split(" ");
                                int suma = 0;
                                for (int k = 0; k < aprecios.length; k++) {
                                        suma = suma + Integer.parseInt(aprecios[k]);
                                }
                                if (barato > suma) {
                                        barato = suma;
                                        mejorOpcion = String.valueOf(j+1);
                                }
                        }
                        aresultados[i] = mejorOpcion;
                }

                for (int i = 0; i < aresultados.length; i++) {
                        System.out.println(aresultados[i]);
                }
        }
}
```

Autor de la solución: Jan Olivetti (Miembro del comité organizador)

La fortuna de Francis

Francis Folk era un prominente programador. Sus habilidades en la computación le habían llevado a ganar grandes cantidades de dinero. No sabiendo qué hacer con su fortuna empezó de bien joven a gastarla en coches que coleccionaba en varios garajes repartidos por Europa y América del norte.
A su muerte su única hija va a heredar toda su fortuna, con una única condición: uno de sus coches, el que ella quiera, debe dárselo a Johnny Deutsch, su archienemigo.

La hija de Francis por supuesto no va a dejar que su orgullo le impida heredar los frutos de la vida de su padre, así que quiere darle a Johnny el peor coche que tenga, pero no entiende de coches. Su mejora amiga le ha sugerido que le dé el más viejo.

Francis empezó a comprar coches al cumplir la mayoría de edad en 2001, y siempre compró coches nuevos y los matriculó en España, donde las matrículas eran cuatro dígitos seguidos de tres letras (las letras tienen prioridad sobre los números, por ejemplo, la matrícula *9999BBB* es anterior a la matrícula *0000ZZZ*).

Francis tiene preparado el listado de todas las matrículas de los vehículos de su padre, ¿puedes decirle cual es la matrícula más antigua?

Input format
Un número *n* seguido de *n* líneas cada una conteniendo una secuencia de 4 dígitos y 3 letras.

Constraints

Output format
La matrícula más antigua.

Solución en Python

```python
def older(a, b):
    if (b == ''): return True
    elif (a[-3:] == b[-3:]): return a[:4] < b[:4]
    else: return a[-3:] < b[-3:]

n = int(input())
best = ''

for i in range(n):
    s = input()

    if (older(s, best)): best = s

print(best)
```

Autor de la solución: Javier López-Contreras (Miembro del comité organizador)

Solución en C++

```cpp
#include <iostream>

using namespace std;

int main()
{
  int n;
  string matricula;
  string matricula_mas_antigua;
  cin >> n;
  cin >> matricula;
  // En ausencia de mas matriculas la primer matricula es la mas antigua
  // Guardamos las matriculas en el siguiente orden: letras, numeros
  // De esta manera podemos compararlas directamente
  matricula_mas_antigua = matricula.substr(4) + matricula.substr(0, 4);
  while (--n) {
    // Para cada matricula miramos si es mas antigua que la mas antigua que hemos
    // visto hasta ahora, y si lo es pasa a ser la mas antigua
    cin >> matricula;
    matricula = matricula.substr(4) + matricula.substr(0, 4);
    if (matricula.compare(matricula_mas_antigua) < 0) {
      matricula_mas_antigua = matricula;
    }
  }
  // Mostramos la matricula mas antigua intercambiando de nuevo letras y digitos
  cout << matricula_mas_antigua.substr(3) + matricula_mas_antigua.substr(0, 3);
}
```

Autor de la solución: Jacobo Vilella (Miembro del comité organizador)

Solución en Java

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Main {

    public static void main(String[] args)
    {
        new Main().start();
    }

    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
    void start()
    {
        try
        {
            System.out.println(realizarCaso());
```

```
    } catch (Exception e){e.printStackTrace();}
}

String realizarCaso() throws Exception
{
    StringBuilder result = new StringBuilder();
    String entrada = reader.readLine();

    char[] matriculaResultLetras = {'Z','Z','Z'};
    char[] matriculaResultNums = {'9','9','9','9'};

    long numMatriculas = 0;

    numMatriculas = Long.parseLong(entrada);


    for (int i = 0; i < numMatriculas; i++)
    {
        char[] matriculaLetras = new char[3];
        char[] matriculaNums = new char[4];

        reader.read(matriculaNums, 0, 4);
        reader.read(matriculaLetras, 0, 3);

        reader.readLine();

        for (int j = 0; j < 7; j++)
        {
            if (j < 3)
            {
                if (matriculaLetras[j] < matriculaResultLetras[j])
                {
                    matriculaResultLetras = matriculaLetras;
                    matriculaResultNums = matriculaNums;

                    break;
                } else if (matriculaLetras[j] > matriculaResultLetras[j])
                {
                    break;
                }
            } else
            {
                if (matriculaNums[j-3] < matriculaResultNums[j-3])
                {
                    matriculaResultLetras = matriculaLetras;
                    matriculaResultNums = matriculaNums;
                    break;
                } else if (matriculaNums[j-3] > matriculaResultNums[j-3])
                {
                    break;
                }
```

```
                }
            }
        }

        for (int i = 0; i < 4; i++)
        {
            result.append(matriculaResultNums[i]);
        }
        for (int i = 0; i < 3; i++)
        {
            result.append(matriculaResultLetras[i]);
        }

        return result.toString();
    }
}
```

Autor de la solución: Daniel López Piris (Concursante OIE 2019)

Flipping rectangles

Santi es un gran amante la hípica y le encanta pasar tiempo a lomos de su caballo Lucy. Sin embargo, últimamente no puede montar a gusto porque sabe que aún no ha conseguido resolver el último juego que le regaló su padre, el flipping rectangles. El juego consiste en un tablero rectangular con casillas blancas y negras, y el objetivo de Santi es que todas la casillas se vuelvan blancas. Para ello, puede realizar movimientos del siguiente tipo: Selecciona una casilla del tablero e invierte todas las casillas del subrectángulo que tiene como esquina superior izquierda la que ha escogido Santi y como casilla inferior derecha la casilla inferior derecha del tablero (invierte cualquier casilla que esté en la misma fila que la que ha escogido o más abajo y en la misma columna que la que ha escogido o más a la derecha). Como Santi quiere volver a montar a su caballo lo antes posible, quiere resolver el juego en el mínimo número de movimientos posible, y os pide que le ayudéis.

Nota: Invertir una casilla significa cambiarla de color. Es decir, si era blanca pasa a ser negra y viceversa.

Ejemplo de movimiento para entender la dinámica del juego:

Si Santi tiene el tablero

1 **0** 1
1 1 0

y selecciona la casilla en negrita, el resultado será el siguiente, con las casillas que han cambiado marcadas en cursiva:

1 *1 0*
1 *0 1*

Input Format

La entrada empieza con un entero *t*, el número de casos. Le siguen los *t* casos, cada uno descrito por dos enteros *n, m* (las dimensiones del tablero) y *n* filas con *m* números cada una, separados por espacios. Estos números describen el tablero y pueden ser *0* si la casilla inicial es blanca o *1* si la casilla inicial es negra.

Constraints

1 <= t <= 40

Caso 2: 19 puntos
1 <= n,m <= 5

Caso 3: 27 puntos
1 <= n,m <= 80

Caso 4: 8 puntos
n = 1,1 <= m <= 1000

Caso 5: 16 puntos
1 <= n,m <= 1000 y el tablero inicial está pintado como uno de ajedrez, alternando los colores blanco y negro de manera que no haya dos casillas adyacentes del mismo color (como en el último ejemplo)

Caso 6: 30 puntos
1 <= n,m <= 1000

Output Format
Imprimid *t* líneas, cada una con un único entero, el número mínimo de movimientos que Santi necesita para resolver el juego.

Solución en Python

```python
ncases = int(input())

for case in range(ncases):
    n, m = input().split(' ')
    n, m = int(n), int(m)

    cols = [0 for i in range(m)]

    res = 0
    for i in range(n):
        cont = 0
        s = input().split(' ')

        for j in range(m):
                x = int(s[j])

                if ((cont + cols[j] + x)%2 == 1):
                        res += 1
                        cont += 1

                cols[j] += cont

    print(res)
```

Autor de la solución: Javier López-Contreras (Miembro del comité organizador)

Solución en C++

```cpp
#include <iostream>
#include <vector>

using namespace std;
```

```cpp
int main() {
  int cas;
  cin >> cas;
  while (cas--) {
    int n, m;
    cin >> n >> m;
    vector<int> V(m, 0);
    int res = 0;
    for (int i = 0; i < n; ++i) {
      int cont = 0;
      for (int j = 0; j < m; ++j) {
        int x;
        cin >> x;
        if ((cont + V[j] + x)%2) {
          ++res;
          ++cont;
        }
        V[j] += cont;
      }
    }
    cout << res << endl;
  }
}
```

Autor de la solución: Jordi Rodríguez (Miembro del comité organizador)

Solución en Java

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Main {

    public static void main(String[] args)
    {
        new Main().start();
    }

    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
    void start()
    {
        try
        {
            int numCasos = Integer.parseInt(reader.readLine());
            for (int casos = 0; casos < numCasos; casos++)
            {
                System.out.println(realizarCaso());
            }

        } catch (Exception e){e.printStackTrace();}
    }
```

```
    int realizarCaso() throws Exception
    {
        int result = 0;
        String entrada = reader.readLine();
        String[] entradaSplit = entrada.split(" ");

        int[][] tablero = new
int[Integer.parseInt(entradaSplit[0])][Integer.parseInt(entradaSplit[1])];
        int[] array = new int[Integer.parseInt(entradaSplit[1])];

        for (int i = 0; i < tablero.length; i++)
        {
            entrada = reader.readLine();
            entradaSplit = entrada.split(" ");
            for (int j = 0; j < tablero[0].length; j++)
            {
                tablero[i][j] = Integer.parseInt(entradaSplit[j]);
            }
        }

        for (int i = 0; i < tablero.length; i++)
        {
            for (int j = 0; j < tablero[0].length; j++)
            {
                If ((tablero[i][j]+array[j])%2 == 1)
                {
                    result++;

                    for (int k = j; k < array.length; k++)
                    {
                        array[k]++;
                    }
                }
            }
        }

        return result;
    }
}
```

Autor de la solución: Daniel López Piris (Concursante OIE 2019)

Professor Oak strikes back

El Profesor Oak está escribiendo un texto en su ordenador y, como es habitual, emplea solamente sus dos dedos índices. El texto en cuestión consta únicamente de letras mayúsculas sin tildes, de los caracteres "," , "." , ":" , ";" , "'" , "?" , "!" y de espacios y saltos de línea. Todos los caracteres, salvo los espacios y los saltos de linea, tienen una tecla asociada, que se muestra en el esquema del teclado a continuación.

Q	W	E	R	T	Y	U	I	O	P	'
A	S	D	F	G	H	J	K	L	;	?
Z	X	C	V	B	N	M	,	.	:	!

El Profesor Oak empieza a escribir teniendo su dedo índice izquierdo sobre la tecla F y su dedo índice derecho sobre la tecla J. En cualquier momento, puede desplazar cualquiera de sus dedos a una de las cuatro teclas adyacentes (o tres o dos si se encuentra en un borde del teclado) y tarda un segundo en hacer este movimiento. Además, es capaz de desplazar ambos dedos simultáneamente. Su prodigiosa habilidad le permite, además, pulsar una tecla instantáneamente (es decir, en cero segundos) si tiene uno de sus dos índices sobre ella. La barra espaciadora y la tecla INTRO, que corresponden a los espacios y los saltos de línea, son muy grandes, de modo que el Profesor Oak las pulsa con la punta de su nariz, también instantáneamente.

Dado el texto que ha escrito el Profesor Oak, determina el tiempo que ha tardado en escribirlo, sabiendo que lo hace de forma óptima. Puedes asumir que el Profesor Oak tiene unos brazos infinitamente flexibles, es decir, que puede entrelazarlos sin problema alguno.

Input Format
Cada caso consiste en un texto, que consta de letras mayúsculas sin tildes, de los caracteres "," , "." , ":" , ";" , "'" , "?" , "!" y de espacios y saltos de línea. El final del texto lo indicará el carácter "-".

Constraints
El texto no tiene más de mil caracteres.

Output Format
Imprime un solo entero: el tiempo que el Profesor Oak ha tardado en escribir el texto.

Solución en C++

```
#include <iostream>
#include <vector>
#include <map>
```

```
using namespace std;

typedef vector<int> VI;
typedef vector<VI> VVI;
typedef vector<VVI> VVVI;
typedef vector<VVVI> VVVVI;
typedef vector<VVVVI> VVVVVI;
typedef map<char,pair<int,int> > MCP;
typedef vector<string> VS;

string s;
VVVVVI V;
MCP M;
VS keys;

const int INF = 1e9;

void init() {
  keys = VS(3);
  keys[0] = {"QWERTYUIOP'"};
  keys[1] = {"ASDFGHJKL;?"};
  keys[2] = {"ZXCVBNM,.:!"};
  for (int i = 0; i < 3; ++i) {
    for (int j = 0; j < 11; ++j) {
      M[keys[i][j]] = {i,j};
    }
  }
}

int f(int a, int b, int c, int d, int e) {
  if (a == s.size()) return 0;
  if (V[a][b][c][d][e] != -1) return V[a][b][c][d][e];
  int sol = INF;
  int dx = abs(b-M[s[a]].first);
  int dy = abs(c-M[s[a]].second);
  for (int i = 0; i < 3; ++i) {
    for (int j = 0; j < 11; ++j) {
      if (abs(d-i) + abs(e-j) <= dx+dy) {
        sol = min(sol,dx+dy+f(a+1,M[s[a]].first,M[s[a]].second,i,j));
      }
    }
  }
  dx = abs(d-M[s[a]].first);
  dy = abs(e-M[s[a]].second);
  for (int i = 0; i < 3; ++i) {
    for (int j = 0; j < 11; ++j) {
      if (abs(b-i) + abs(c-j) <= dx+dy) {
        sol = min(sol,dx+dy+f(a+1,i,j,M[s[a]].first,M[s[a]].second));
      }
    }
  }
  return V[a][b][c][d][e] = sol;
}
```

```
int main() {
  init();
  int T;
  cin >> T;
  while (T--) {
  s = "";
  char t;
  while (cin >> t and t != '-') s += t;
    V = VVVVVI(s.size(), VVVVI(3,VVVI(11,VVI(3,VI(11,-1)))));
    cout << f(0,1,3,1,6) << endl;
  }
}
```

Autor de la solución: Jordi Castellví (Miembro del comité organizador)