Ejercicio 1

2 puntos

Las convenciones de código son reglas y estándares que guían la escritura de programas para mejorar la legibilidad y consistencia del código. Estas convenciones incluyen aspectos como el formato del código, la nomenclatura de variables, funciones y clases, el uso de comentarios y la organización del código. Seguir estas convenciones no solo facilita la comprensión del código por parte de otros programadores, sino que también ayuda a prevenir errores y a mantener un código más ordenado a lo largo del tiempo.

Con respecto al formato, existen distintas convenciones en lo que respecta a la *indentación* del código, es decir, la práctica de insertar espacios al comienzo de cada línea para estructurar visualmente el programa. En el lenguaje C, la indentación está ligada a los bloques de código, que se delimitan entre llaves. Sin embargo, según el estilo utilizado, las llaves se colocan de una u otra manera. Por ejemplo:

Estilo Kernighan y Ritchie

```
while (x == y) {
  f(x);
}
```

Estilo Allman

```
while (x == y)
{
  f(x);
}
```

Estilo Whitesmiths

```
while (x == y)
  {
  f(x);
  }
```

El estilo *Whitesmiths* coloca las llaves de apertura y cierre en líneas separadas. La llave de apertura `'{'` se sitúa una o más posiciones a la derecha con respecto a la línea anterior. El resto de sentencias del bloque, incluyendo la llave de cierre `'}'`, tienen la misma indentación que la llave de apertura. Por ejemplo, el siguiente fragmento de código sigue este estilo de indentación, donde los puntos (`·`) denotan espacios en blanco.

```
void mi_funcion(int x)   // Sin indentacion: 0 espacios
··{                      // Indentacion: 2 espacios
··if (x > 0)             // Indentacion: 2 espacios
····{                    // Indentacion: 6 espacios
····f(x);                // Indentacion: 6 espacios
····}                    // Indentacion: 6 espacios
··}                      // Indentacion: 2 espacios
```

Ten en cuenta que la diferencia (en número de espacios) entre un nivel de indentación y el siguiente puede variar. Lo importante es que un nivel de indentación mayor tiene más espacios al principio de la línea que un nivel menor.

El objetivo de este ejercicio es determinar si un programa en C está correctamente indentado según el estilo Whitesmiths. Para ello basta con conocer el primer carácter no blanco de cada línea, y saber cuántos espacios tiene delante. Por tanto, supongamos una lista con tantos elementos como líneas tiene el programa. Cada elemento es un par *(n, c)*, donde *c* contiene el primer carácter no blanco de la línea y *n* es el número de espacios que hay antes de *c*. Por ejemplo, el programa anterior puede representarse mediante la lista de pares `[(0, 'v'), (2, '{'), (2, 'i'), (6, '{'), (6, 'f'), (6, '}'), (2, '}')]`. Por simplicidad, puedes suponer que el programa no tiene líneas en blanco, y que la primera línea del programa no contiene una llave de apertura.

Escribe una función con la siguiente cabecera:

```
int comprobar_whitesmiths(const list<pair<int, char>> &programa);
```

La función debe devolver -1 si la indentación del programa pasado como parámetro sigue el estilo Whitesmiths. En caso contrario, debe devolver un valor entero indicando el número de la primera línea que no está correctamente indentada. Puedes suponer que el programa de entrada es sintácticamente correcto (es decir, las llaves están correctamente equilibradas) y que cada una de las llaves del programa, tanto de apertura como de cierre, están en una línea propia.

Indica y justifica el coste de `comprobar_whitesmiths` en función del tamaño de la lista de entrada.

Entrada

La entrada consta de varios casos de prueba. Cada uno de ellos consiste en un número *N* que indica el número de líneas del programa. Después vienen *N* líneas con el texto del programa. Suponemos que se utilizan espacios en blanco (y no tabuladores) para la indentación.

La entrada finaliza con un programa de 0 líneas, que no se procesa.

Salida

Para cada caso de prueba debe imprimirse la palabra `CORRECTO` si el programa de entrada sigue la convención de estilo Whitesmiths para la indentación. En caso contrario, debe imprimirse el número de la primera línea que no está correctamente indentada. Las líneas se numeran comenzando desde el 1.

Entrada de ejemplo

```
11
#include <stdio.h>
int main()
  {
  int x;
  scanf("%d", &x);
  if (x > 0)
       {
       printf("Valor positivo\n");
       }
  return 0;
  }
4
int main()
{         // <- Esta linea y las dos siguientes deben indentarse
int x;
}
1
  const int x = 3;  // <- El nivel de indentacion inicial debe ser 0
5
int main()
  {
    int x;  // <- La indentacion debe ser la misma que la de la llave '{'
    x = 3;
  }
0
```

Salida de ejemplo

```
CORRECTO
2
1
3
```