Reconocimiento de caracteres

Este problema requiere escribir un programa que reconozca caracteres.

Detalles:

Cada imagen ideal de un carácter tiene 20 líneas de 20 dígitos. Cada dígito es un ´0´ o un ´1´. En la ***Figura 1A*** hay un ejemplo de representación de las imágenes de los caracteres del archivo.

El archivo FONT.IN contiene las representaciones de 27 imágenes ideales de caracteres en el siguiente orden:

□abcdefghijklmnopqrstuvwxyz

donde □ representa el espacio en blanco.

El archivo IMAGE.IN contiene una o más imágenes de caracteres que pueden estar corruptas. Una imagen de carácter puede estar corrupta de las siguientes maneras:

- Como máximo una línea puede estar duplicada (la original y la duplicada están contiguas).
- Como máximo ha desaparecido una línea.
- Algunos '0' (ceros) están cambiados por '1' (unos).
- Algunos '1' (unos) están cambiados por '0' (ceros).

Ninguna imagen de carácter tiene simultáneamente una línea duplicada **y** una línea desaparecida. En cada imagen de un carácter de los juegos de prueba de la evaluación no habrá más de un 30% de dígitos cambiados.

En el caso de una línea duplicada, una o ambas pueden tener datos corruptos y estos pueden ser diferentes.

Tarea:

Escribe un programa que reconozca la secuencia de una o varias imágenes de caracteres contenida en el archivo IMAGE.IN usando la fuente contenida en el archivo FONT.IN.

Se reconoce una imagen de un carácter eligiendo la imagen del carácter fuente que requiere el menor número de cambios de dígitos para transformarla en la que queremos reconocer, tomando las decisiones más favorables sobre líneas duplicadas o desaparecidas. En el caso de una línea duplicada sólo contar las corrupciones de la línea menos corrupta. Todos los caracteres de la imagen del ejemplo y de los juegos de prueba para evaluación son reconocibles uno a uno por un programa correcto. Solo hay una única solución óptima para cada ejemplo de la prueba de evaluación.

Precisamente, una solución correcta usará todos los datos indicados en el archivo de entrada IMAGE.IN.

Datos de entrada:

Ambos archivos de entrada empiezan en un entero N ($19 \le N \le 1200$) que especifica el número de líneas que le siguen:

N
(dígito1)(dígito2)(dígito3) … (dígito20)
(dígito1)(dígito2)(dígito3) … (dígito20)
…

Cada línea de datos contiene 20 dígitos. No hay espacios en blanco entre los dígitos.

El archivo FONT.IN describe a la fuente. FONT.IN siempre consistirá en 541 líneas. FONT.IN puede ser diferente en cada juego de pruebas.

Datos de salida:

El programa debe generar el archivo IMAGE.OUT, que contiene un sólo string (cadena de caracteres) con los caracteres reconocidos. Su formato es una sencilla línea de texto en ASCII. La salida no debe contener ningún separador de caracteres. Si tu programa no reconoce a un carácter en particular debe mostrar un '?' en la posición adecuada.

Precaución: el formato de salida antes especificado anula el formato de salida estándar especificado en las reglas generales, que requiere espacios separadores en la salida.

Puntuación:

Vendrá dada por el porcentaje de caracteres reconocidos correctamente.

VEA LOS EJEMPLOS EN LA OTRA CARA.

Ejemplo de archivos:

Ejemplo incompleto mostrando el principio de FONT.IN (espacio en blanco y ‘a’).

FONT.IN

```
540
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000111100000000000
00001111111011000000
00001111111001100000
00001111000110010000
00001100001100010000
00001100000100010000
00000100000100010000
00000010000000110000
00000001000001110000
00001111111111110000
00001111111111110000
00001111111111000000
00001000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
```

Figura 1A

Ejemplo mostrando una ‘a’ corrupta.

IMAGE.IN

```
19
00000000000000000000
00000000000000000000
00000000000000000000
00000011100000000000
00100111011011000000
00001111111001100000
00001110001100100000
00001100001100010000
00001100000100010000
00000100000100010000
00000010000000110000
00001111011111110000
00001111111111110000
00001111111111000000
00001000010000000000
00000000000000000000
00000000000001000000
00000000000000000000
00000000000000000000
```

Figura 1B

Ejemplo de salida:

IMAGE.OUT	Explicación
a	Carácter ‘a’ reconocido

Figura 2