Double Crypt

PROBLEM

The Advanced Encryption Standard (AES) involves a new strong encryption algorithm. It works with three *blocks* of 128 bits. Given a message block p (plaintext) and a key block k, the AES encryption function E returns an encrypted block c (ciphertext):

$$c = E(p, k) .$$

The inverse of the AES encryption function E is the decryption function D such that

$$D (E(p, k), k) = p , \quad E (D(c, k), k) = c .$$

In *Double AES*, two independent key blocks k_1 and k_2 are used in succession, first k_1, then k_2:

$$c_2 = E (E(p, k_1), k_2).$$

In this task, an integer s is also given. Only the leftmost $4*s$ bits of all keys are relevant, while the other bits (the rightmost 128 minus $4*s$ bits) are all zero.

You are to recover the encryption key pairs for some messages encrypted by Double AES. You are given both the plaintext p and the corresponding double-encrypted ciphertext c_2, and the structure of the encryption keys as expressed by the integer s.

The AES encryption and decryption algorithms are available in a library.
You must submit the recovered keys, and not a recovery program.

INPUT

You are given ten problem instances in the text files named `double1.in` to `double10.in`. Each input file consists of three lines. The first line contains the integer s, the second line the plaintext block p, and the third line the ciphertext block c_2 obtained from p by Double AES encryption. Both blocks are written as strings of 32 hexadecimal digits ('0'..'9', 'A'..'F'). The library provides a routine to convert strings to blocks. All input files are solvable.

OUTPUT

You are to submit ten output files corresponding to the given input files. Each output file consists of three lines. The first line contains the text
`#FILE double I`
where `I` is the number of the respective input file. The second line contains the key block k_1, and the third line the key block k_2, such that

$$c_2 = E (E(p, k_1), k_2).$$

Both blocks must be written as strings of 32 hexadecimal digits ('0'..'9', 'A'..'F'). The library provides a routine to convert blocks to strings. If there are multiple solutions, you need submit only one of them.

EXAMPLE

As an example we use input file number 0 here.

`double0.in`

```
1
00112233445566778899AABBCCDDEEFF
6323B4A5BC16C479ED6D94F5B58FF0C2
```

A possible output file

```
#FILE double 0
A0000000000000000000000000000000
70000000000000000000000000000000
```

LIBRARY

FreePascal library (Linux: `aeslibp.p, aeslibp.ppu, aeslibp.o;` Windows: `aeslibp.p, aeslibp.ppw, aeslibp.ow`):

```
type
  HexStr = String [ 32 ]; { only '0'..'9', 'A'..'F' }
  Block  = array [ 0..15 ] of Byte; { 128 bits }

procedure HexStrToBlock ( const hs: HexStr; var b: Block );
procedure BlockToHexStr ( const b: Block; var hs: HexStr );
procedure Encrypt ( const p, k: Block; var c: Block );
  { c = E(p,k) }
procedure Decrypt ( const c, k: Block; var p: Block );
  { p = D(c,k) }
```

The program `aestoolp.pas` illustrates how to use the FreePascal library.

GNU C/C++ library (Linux and Windows: `aeslibc.h, aeslibc.o`):

```
typedef char HexStr[33]; /* '0'..'9', 'A'..'F', '\0'-terminated */
typedef unsigned char Block[16];  /* 128 bits */

void hexstr2block ( const HexStr hs, /* out-param */ Block b );
void block2hexstr ( const Block b, /* out-param */ HexStr hs );
void encrypt ( const Block p, const Block k, /* out-param */ Block c );
  /* c = E(p,k) */
void decrypt ( const Block c, const Block k, /* out-param */ Block p );
  /* p = D(c,k) */
```

The program `aestoolc.c` illustrates how to use the GNU C/C++ library.

CONSTRAINTS

For the number s of relevant hexadecimal digits in a key it holds that $1 \le s \le 5$.

HINT: A good program can recover keys in less than 10 seconds for any allowed input file.