**Encryption**

A block cipher method called chaining can be used to make a much more secure ciphertext message. In this problem you will use the cipher block chaining method to encrypt a message.

Here are the steps you should follow:

1. Convert the ascii key to a binary (10101010) representation.
2. Convert the ascii text to binary representation.
3. Break the binary text from number 2 up into some larger blocks (you will use 12 bits as your block size, but you theoretically could use anything). If the last block is less than 12, you should add alternating 1’s and 0’s to it until you have 12. Finally, if you added some padding, you should append a final block (of size 12) that contains the number of bits you added. I.e. If I added 4 bits of padding, I would append the block ‘00000000100’ as the final block to be encoded.
4. Each of the above (12 bit) blocks will now be encrypted, following this process:
   - Take the first block to be encrypted, reverse it, Xor it with the first 12 bits of the key.
   - Take the next block and Xor it with the encrypted output of the first block. Reverse the new block, Xor it with the first 12 bits of the key.
   - Repeat until all the blocks are encrypted.

Your code should take an input file with the ASCII key on the first line, the text to be encoded on the remaining lines, convert them to their binary representation, and use the cipher block chaining method as described above to ‘encrypt’ them. The resultant bitstream should then be output.

**Example:**

This input:

```
ABC ABCDE
```

Should produce this output:

```
0110100101101110101001100000011000011001001011100010010111101
```

**Hints:**

```
ABCDE in binary:
0100000101000010010000110100010001000101

After padding is added (to get last block up to 12 bits):
010000010100001001000011010001000100010110101010

After the block is added that tells us how much padding was added (12-bit block):
01000001010000100100001101000100010001011010101000000000100

Key in binary (first 12 bits): 010000010100 First 12 bit block (reversed): 001010000010 Xor : 0110100101101 #first 12 bits of encrypted output

2nd 12 bit block : 001001000011 Output of last Xor : 011010010110 Xor : 010011010101

Key in binary (first 12 bits): 010000010100 Xor’ed output (reversed) : 101010110010 Xor : 111010100110 #next 12 bits of encrypted output

2nd 12 bit block : 010001000010 Output of last Xor : 111010100110 Xor : 101011100010

And so on...