```python
import math
import itertools

################################################################################
#PROBLEM 1
################################################################################
#
#INPUT positive number n
#RETURN log of number base 2
def log_2(n):
    pass

#INPUT list of immutable objects
#RETURN probability distribution
def makeProbability(xlst):
    pass

#INPUT probability distribution
#RETURN non-negative number entropy
def entropy(xlst):
    pass



################################################################################
#PROBLEM 2
################################################################################
#INPUT positive integer
#RETURN positive integer
def magick(x):
    pass


################################################################################
#PROBLEM 3
################################################################################
#INPUT a list of lists of three positive integers [[a,b,c],[d,e,f],[g,h,i]]
```

```python
#RETURN True if the input is a magic square
#You can create other functions to help you--they will
#not be unit tested
def is_magic_square(s3):
    pass



#INPUT nothing
#RETURN list of solutions to magic square size 3
def generate_3_square():
    pass



################################################################################
# PROBLEM 4
################################################################################

#INPUT takes a letter and shift
#RETURN new letter shifted
def encrypt(letter, n):
    pass

#INPUT takes a letter and shift
#RETURN original letter
def decrypt(letter, n):
    pass
#INPUT takes a sentence of lowercase letters and spaces and shift
#RETURN caeser cypher
def encrypt_sentence(sentence, shift):
    pass

#INPUT takes an encrypted sentence and shift
#RETURN decrypted sentence
def decrypt_sentence(sentence, shift):
    pass
```

```
#############################################################################
# PROBLEM 5
#############################################################################

#INPUT non-negative integer and non-negative integer > 1
#RETURN Wild Number [string, base]
#string is encoding of number in base, base is integer
def make_number(decimal, base):
    pass

#INPUT Wild number
#RETURN new wild number in new base
def convert(number, base):
    pass

#INPUT two wild numbers
#RETURN product as a (possibly new) base
def mul_(number1, number2, base):
    pass

#INPUT two wild numbers
#RETURN sum as a (possibly new) base
def add_(number1, number2, base):
    pass
    protein =
translate(DNA_d)#############################################################################

# Problem 6

#INPUT path to amino acid file
#RETURN a dictionary
#Key is a tuple (c0, c1, ... , cn) where ci are codons
#Value is a pair [name, abbreviation] for the amino acid
def get_amino_acids(file_path):
    pass
```

```
#INPUT path to DNA file
#RETURN a list [header, DNA]
#header is first line in the file
#DNA is a string of letters from remainder of file
#no whitespace
def get_DNA(file_path):
    pass

#INPUT FAST file
#RETURN a string representing the protein
#using the dictionary
def translate(DNA_d):
    pass
```