Write your name at the top of this page. Do not write your name on any other pages.

You have 70 minutes. Notes, books, computers, etc. are forbidden.

Show your work. Incorrect or incomplete answers with good work often earn partial credit. Correct answers with insufficient explanation might not earn full credit.

If you are asked to write code but you do not know the exact Python required, then try to write code that is approximately correct. If you think that your code does not demonstrate that you understand the solution, then describe your idea in English as well. Be precise enough that I cannot misinterpret your solution.

Good luck!

```
def frequencies(string):
    counts = [0] * 26
    for char in string:
        index = indexFromLetter(char)
        if index != None:
            counts[index] += 1
    total = sum(counts)
    freqs = [0.0] * 26
    for i in range(len(counts)):
        freqs[i] = counts[i] / total
    return freqs
```

The code above is part of the frequency analysis in ciphers.py. For each variable or expression listed below, give its type. If it is a list, then also give the type of the elements in that list.

A1. `frequencies`

A2. `string`

A3. `counts`

A4. `char`

A5. `index`

A6. `index != None`

A7. `counts[index]`

A8. `total`

A9. `sum`

A10. `freqs`

A11. `i`

A12. `range(len(counts))`

B. Describe the color orange in RGB. First give your answer such that each channel is measured on a scale from 0 to 255. Then give your answer in 24 bits. Then give your answer in hexadecimal. Your three answers should exactly agree.

In this course we have written a function `isPrime` to test whether a given integer n is prime, and we have modified `isPrime` into a function `factor` that returns the smallest factor of n.

C. Write `factor` in Python. [My solution is 5 lines of code.]

In the USA, a standard deck of playing cards consists of 52 cards, organized into 4 *suits* of 13 *kinds*. The four suits are called *spades*, *hearts*, *diamonds*, and *clubs*. The 13 kinds are called 2, 3, 4, 5, 6, 7, 8, 9, 10, *jack*, *queen*, *king*, and *ace*. A *hand* is a set of 5 distinct cards. A hand has *three of a kind* if three (or more) of its cards are of the same kind.

Suppose that you're writing a card game in Python. You decide to represent a card as a pair of strings, such as `['6', 'spades']` or `['queen', 'diamonds']`, and a hand as a length-5 list of cards. You want to write a function, `hasThreeOfAKind`. As input it takes a hand. As output it returns a Boolean indicating whether the hand has three of a kind.

D1. Describe in English your idea for implementing `hasThreeOfAKind`. Be precise. For example, any loops should be clearly indicated.

D2. Write `hasThreeOfAKind` in Python. [My solution is 9 or 10 lines of code.]

E1. Give the 8-bit two's complement representations of 12 and -10.

E2. Multiply those two numbers using the usual algorithm. Include any overflow that occurs.

E3. Does this multiplication give the correct result? Explain?

Suppose that Alice sets up an RSA cryptosystem consisting of n (public), e (public), and d (private). As we have seen, this system lets people send Alice encrypted messages. But RSA can also be used for an entirely different purpose: *authentication*. Suppose that Babatope is communicating with someone over the Internet. She claims to be Alice, but he is worried that she is not really Alice. So Babatope sends this "Alice" a piece of information, called the *challenge*. "Alice" does a computation and sends another piece of information back, called the *response*. Babatope does another computation. Based on the result, Babatope decides whether "Alice" is really Alice, the owner of d.

F. Fill in the details. What are the challenge and response? How does Babatope decide?