“My CS 5 Black Lecture Notes Were Stolen By Aliens” claims distraught CS 5 Professor.

Claremont, CA: A Harvey Mudd CS Professor claims that his lecture notes were stolen by five-eyed aliens early on Thursday morning. “I wrote up some notes for my lecture and, while I went out to get a Spam sandwich before class, aliens snuck into my office and stole my notes to eat for breakfast,” claimed the distraught professor. “We’re obligated to investigate every claim,” said a campus security officer, “but we’re skeptical of this one. Aliens are not known to eat class notes for breakfast. They much prefer Spam-flavored Pop Tarts and Red Bull. We’re investigating, but we suspect that squirrels were the actual perpetrators of this terrible crime.” Why are you reading this drivel? Don’t you have anything better to do?!
True Story...

That’s crArizonay!
Date: Wed, 10 Feb 2010 10:39:45 -0800 (PST)
From: Arthur Benjamin <benjamin@math.hmc.edu>
To: Tina Straley <tstraley@maa.org>
Cc: Arthur Benjamin <benjamin@math.hmc.edu>
Subject: amusing (?) typo

Dear Tina (or should that be TIndianaa?):

Yesterday I received the attached letter from MAA membership with my "new" name, "Arthur BeNew Jerseyyamin". Apparently the NJ in the middle of my last name was replaced by New Jersey.

I laughed it off, but I worry that this could be a bigger problem, and thought you should know.

Art (or should that be Arkansast?)
I’m away tomorrow...
ED: Edit Distance

```python
>>> ED("qspam", "scramble")
6

qspam -> (delete q)
spam -> (substitute p with c)
scam -> (insert r)
scram -> (insert b)
scramb -> (insert l)
scrambl -> (insert e)
scramble
```

```python
def ED(S1, S2):
    if S1 == ":
        return len(S2)
    elif S2 == ":
        return len(S1)
    elif S1[0] == S2[0]:
        return ED(S1[1:], S2[1:])
    else:  # substitute, insert, or delete!
        substitute =
        insert =
        delete =
        return min(substitute, insert, delete)
```

Permitted “morphing” operations:
- insert
- delete
- substitute

It’s safe to choose to morph the string left-to-right!

Fill this in your notes

6 steps
ED: Edit Distance

```python
>>> ED("qspam", "scramble")
6

qspam -> (delete q)
spam -> (substitute p with c)
scam -> (insert r)
scram -> (insert b)
scramb -> (insert l)
scrambl -> (insert e)
scramble
```

```python
def ED(S1, S2):
    if S1 == ": return len(S2)
elif S2 == ": return len(S1)
elif S1[0] == S2[0]: return ED(S1[1:], S2[1:])
else:  # substitute, insert, or delete!
    substitute = 1 + ED(S1[1:], S2[1:])
    insert = 1 + ED(S1, S2[1:])
    delete = 1 + ED(S1[1:], S2)
    return min(substitute, insert, delete)
```

This is scary slow! But, you’ll memoize this to make it fast!
def square(X):
    return X**2

>>> list( map(square, [1, 2, 3, 4, 5]) )
[1, 4, 9, 16, 25]

>>> list( map(lambda X: X**2, [1, 2, 3, 4, 5]) )
[1, 4, 9, 16, 25]

>>> f = lambda X: X**2
>>> f(2)
4

>>> g = lambda X, Y: X*Y
>>> g(2, 3)
6

In Python, lambda expressions can only compute very simple one-line functions (e.g., no conditional statements, etc.). In many programming languages, lambda expressions can be as complex as any function!
increment

>>> increment([10, 20, 30])
[11, 21, 31]

def increment(myList):
    return
increment

>>> increment([10, 20, 30])
[11, 21, 31]

def increment(myList):
    return list(map(lambda X: X+1, myList))
Power Set!

```python
>>> powerset([2, 3])
[[], [3], [2], [2, 3]]
```

```python
>>> powerset([1, 2, 3])
[[], [3], [2], [2, 3], [1], [1, 3], [1, 2], [1, 2, 3]]
```

```python
>>> powerset([1])
[[], [1]]
```

```python
>>> powerset([])
???
```

This really demonstrates the power of functional programming!

Note that the order within each sublist is important, but the order in which the sublists are presented is not!
def powerset(myList):
    if myList == []: return ______
    else:
        loseIt = powerset(myList[1:])
        useIt = list(map(lambda X: [myList[0]] + X, loseIt))
        return loseIt + useIt
def powerset(myList):
    if myList == []: return [[[]]]
    else:
        loseIt = powerset(myList[1:])
        useIt = list(
            map(lambda X: [myList[0]]+X, loseIt))
        return loseIt + useIt
Map of Shmorbodia!
(Assume all “hops” are to the east)

>>> sp(0, 4, shmorbodia)
10

>>> sp(1, 3, shmborbodia)
6
from math import *  # inf is defined here

shmorbodia = [
    [0, 1, 3, 7, inf],
    [inf, 0, 42, 6, 27],
    [inf, inf, 0, 2, 13],
    [inf, inf, inf, 0, 5],
    [inf, inf, inf, inf, 0]
]
A quick aside about the import business!

from math import *
x = factorial(42)
foo = cos(x)
bar = pi/foo

import math
x = math.factorial(42)
foo = math.cos(x)
bar = math.pi/foo

How about a quick aside about foo and bar?
from math import *  # inf is defined here

shmorbodia = [[0, 1, 3, 7, inf],
              [inf, 0, 42, 6, 27],
              [inf, inf, 0, 2, 13],
              [inf, inf, inf, 0, 5],
              [inf, inf, inf, inf, 0]]

>>> shmorbodia[0][3]
def sp(start, end, distInfo):
    if start == end: return ____
    elif start > end: return ____
    else:
        options = map(
            lambda X: distances[start][X] +
                    sp(X, end, distInfo),
            range(start+1, end+1)
        )
    return min(options)
def sp(start, end, distInfo):
    if start == end: return 0
    elif start > end: return inf
    else:
        options = map(
            lambda X: distInfo[start][X] +
                       sp(X, end, distInfo),
            range(start+1, end+1))
        return min(options)

>>> sp(0, 4, shmorbodia)
map(
    lambda X: shmorbodia[0][X] +
              sp(X, 4, shmorbodia),
    range(1,5) )
[1, 2, 3, 4]
def f1(x):
    return 5 * x + 1

def f2(x):
    return x ** 2

>>> newf = upper(f1, f2)

>>> newf(1)

>>> newf(2)

>>> newf(10)

def upper(f1, f2):
    return lambda x: max(f1(x), f2(x))

What happens if we run upper(42, “spam”) ???

upper takes functions as input and returns a function as output!
Super Envelope!

```python
def times10(x):
    return 10 * x

def add42(x):
    return x + 42

def ttt(x):
    return 2 ** x

flist = [times10, add42, ttt]  # Wow! A list of functions!

>>> f = se(flist)
>>> f(1)
>>> f(2)
>>> f(5)
>>> f(10)

def se(funcList):
    return
```

In your worksheet...
def times10(x):
    return 10 * x

def add42(x):
    return x + 42

def ttt(x):
    return 2 ** x

flist = [times10, add42, ttt]

>>> f = se(flist)
>>> f(1)
>>> f(2)
>>> f(5)
>>> f(10)

def se(funcList):
    return lambda x: max(map(lambda f: f(x), funcList))