L5 PROBLEM 8 (5 points possible)

We can use the idea of **bisection search** to determine if a character is in a string, so long as the string is sorted in alphabetical order.

First, test the middle character of a string against the character you're looking for (the "test character"). If they are the same, we are done - we've found the character we're looking for!

If they're not the same, check if the test character is "smaller" than the middle character. If so, we need only consider the lower half of the string; otherwise, we only consider the upper half of the string. (Note that you can compare characters using Python's `<` function.)

Implement the function `isIn(char, aStr)` which implements the above idea recursively to test if `char` is in `aStr`. `char` will be a single character and `aStr` will be a string that is in alphabetical order. The function should return a boolean value.

As you design the function, think very carefully about what the base cases should be.

```
def isIn(char, aStr):
    '''
    char: a single character
    aStr: an alphabetized string

    returns: True if char is in aStr; False otherwise
    '''
    # Your code here

```

Unanswered

Note: In programming there are many ways to solve a problem. For your code to check correctly here, though, you must write your recursive function such that you make a recursive call directly to the function `isIn`. Thank you for understanding.

Hints
Basic function structuring
What should your base case be?
What should your recursive case be?

Check

Show Discussion

New Post

EdX is a non-profit created by founding partners Harvard and MIT whose mission is to bring the best of higher education to students of all ages anywhere in the world, wherever there is Internet access. EdX's free online MOOCs are interactive and subjects include computer science, public health, and artificial intelligence.