```
input_str_1 = "lucidProgramming"
input_str_2 = "LucidProgramming"
input_str_3 = "lucidprogramming"


def find_uppercase_iterative(input_string):
    for i in range(len(input_string)):
        if input_string[i].isupper():
            return input_string[i]
    return "No uppercase character found"


def find_uppercase_recursive(input_string, idx=0):
    if input_string[idx].isupper():
        return input_string[idx]
    if idx == len(input_string) - 1:
        return "No uppercase character found"
    return find_uppercase_recursive(input_string, idx + 1)


def find_length_string_iterative(input_str):
    i = 0
    count = 0
    for s in input_str:
        count += 1
    return count


print(find_length_string_iterative(input_str_1))


def find_length_string_recursive(input_str):
    if input_str == '':
        return 0
    return 1 + find_length_string_recursive(input_str[1:])


print(find_length_string_recursive(input_str_1))

inp_1 = "abc de"
inp_2 = "LuCiDProGrAmMiNG"
vowel = "aeiou"


def count_consonants_iterative(input_str):
    count = 0
    for i in range(len(input_str)):
        if input_str[i].lower() not in vowel and input_str[i].isalpha():
            count += 1
    return count


def count_consonants_recursive(input_str):
    if input_str == '':
        return 0
    if input_str[0].lower() not in vowel and input_str[0].isalpha():
        return 1 + count_consonants_recursive(input_str[1:])
    else:
        return count_consonants_recursive(input_str[1:])


print(count_consonants_iterative(inp_1))
print(count_consonants_recursive(inp_1))

x = 5
y = 3
```

```
def recursive_multiply(x, y):
    # This cuts down on the total number of recursive calls
    if x < y:
        return recursive_multiply(y, x)

    if y == 0:
        return 0
    return x + recursive_multiply(x, y - 1)


print(x * y)
print(recursive_multiply(x, y))
```