National University of Singapore
School of Computing
CS1010S: Programming Methodology

Extra Practice 3 Solutions

Help

Yes, we put these functions here so that you can refer to them easily.

```
def sum(term, a, next, b):
    if a > b:
        return 0
    else:
        return term(a) + sum(term, next(a), next, b)

def fold(op, f, n):
    if n == 0:
        return f(0)
    else:
        return op(f(n), fold(op, f, n-1))
```

Order of Growth

Determine the time and space complexity of all these functions.

(a)
```
def lol1(n, m):
    result = 0
    for i in range(n):
        for j in range(m):
            result += 1
    return result
# Time: O(nm), Space: O(1)
```

(b)
```
def lol2(n):
    result = 0
    for i in range(n):
        for j in range(n):
            result += 1
    return result
# Time: O(n**2), Space: O(1)
```

(c)
```
def lol3(n):
    result = ''
    for i in range(n):
        result += 'a'
    return result
# Time: O(n**2), Space: O(n)
```

(d)
```
def lol4(n):
    if n == 0:
        return 0
    else:
        return lol4(n - 1)
# Time: O(n), Space: O(n)
```

(e)
```
def lol5(n):
    result = 0
    for i in range(n):
        for j in range(i, n):
            result += 1
    return result
# Time: O(n**2), Space: O(1)
```

(f)
```
def lol6(n):
    if n >= 1:
        return 0
    print("CS1010S is fun!")
    lol6(n // 2)
    lol6(n // 2)
# Time: O(1), Space: O(1)
```

(g)
```
def lol7(n):
    for i in range(n):
        for j in range(n + 1, i):
            print("Hello, I am Baymax")
# Time: O(n), Space: O(1)
```

(h)
```
def lol8(n):
    if n < 2:
        print("Less than two")
        return 1
    else:
        for j in range(1,n):
            print("CS1010S is fun!")
        a = lol8(n // 3)
        b = lol8(n // 3)
        c = lol8(n // 3)
        return a + b + c
# Time: O(n log n), Space: O(log n)
```

(i)
```
def lol9(n):
    if n <= 1:
        return
    print("CS1010S")
    for i in range(1, 2):
        lol9(n - 1)
# Time: O(n), Space: O(n)
```

Higher Order Functions

(a) Define a function `total` that produces the output of the following code using either `sum` or `fold`.

2 + 4 + 6 + 8 + 10

Solution:

```
# There are a lot of possibilities, here are the frequently used ones.
def total():
    return sum(lambda x: x, 2, lambda x: x + 2, 10)

def total():
    return sum(lambda x: 2*x, 1, lambda x: x + 1, 5)

def total():
    return fold(lambda x, y: x + y, lambda x: 2*x, 5)

def total():
    return fold(lambda x, y: 2*x + y, lambda x: x, 5)

def total():
    return fold(lambda x, y: x + y, lambda x: 2*x + 2, 5)

def total():
    return fold(lambda x, y: 2*x + y, lambda x: x + 1, 5)
```

(b) I would like to convert a password such as "orange" into a string that comprises **only** of "*", depending on how long my word is. This function will be named `convert` and take in a word string as an input while returning the converted word. You may assume that the word will be at least one letter long.
Sample Output:

```
>>> convert("orange")
'******'
>>> convert("ap13")
'****'
```

- Use an iterative approach to solve this. What is the time and space complexity?
  Solution:

```
# Actually Extra Practice 2
def convert(word):
    answer = ""
    for i in word:
        answer += "*"
    return answer

# Time: O(n**2) time complexity due to string concatenation
# Space: O(n) where n is the length of the word
```

- Use a recursive approach to solve this. What is the time and space complexity?
  Solution:

```
def convert(word):
    if len(word) == 1:
        return "*"
    return "*" + convert(word[1:])
    # or return convert(word[:-1]) + "*"

# Time: O(n**2) due to string concatenation
# Space: O(n**2) where n is the length of the word
# (check recursion tree)
```

- Use the `fold` function to solve this.
  Solution:

```
def convert(word):
    return fold(lambda x, y: x + y, lambda x: "*", len(word) - 1)
```

- Explain if the `sum` function can be used to solve this. If not, explain what change needs to be made to the original function and define it in terms of `sum`.
  Solution:
  Note that the base case of the function is 0 but we are to return a string, so one suggestion is to change 0 on the base case to an empty string then we can do similar to what we did to our solution using `fold`.

(c) Now, I would like to filter out the letters "o" and "a" because I don't really like them. Define a function `remove` that takes in a word and returns the new word with all the "o"s and "a"s removed.

```
>>> remove("orange")
'rnge'
>>> remove("oooaaat")
't'
```

- Use an iterative approach to solve this. What is the time and space complexity?
  Solution:

```
def remove(word):
    answer = ""
    for i in word:
        if i != "o" and i != "a":
            answer += i
    return answer
```

- Use a recursive approach to solve this. What is the time and space complexity?
  Solution:

```
def remove(word):
    if len(word) == 1:
        if word[0] == "o" or word[0] == "a":
            return ""
        return word
    if word[0] == "o" or word[0] == "a":
        return remove(word[1:])
    return word[0] + remove(word[1:])
```

- Use the `fold` function to solve this.
  Solution:

```
def remove(word):
    def is_oa(letter):
        return letter == "o" or letter == "a"
    return fold(lambda x, y: x + y,
                lambda x: "" if is_oa(word[-1-x]) else word[-1-x],
                len(word) - 1)
    # You can change the indexing from -1-x to len(word)-1-x
    # which makes more sense but longer to write
```

- Explain if the `sum` function can be used to solve this. If not, explain what change needs to be made to the original function and define it in terms of `sum`.
  Solution:
  Similar to above, note that the base case of the function is 0 but we are to return a string, so one suggestion is to change 0 on the base case to an empty string then we can do similar to what we did to our solution using `fold`.

- ***How do you modify the functions to remove all vowels in a word?***
  Solution:
  Simply add more conditionals such as

```
i != "o" and i != "a" and i != "e" and i != "i" and i != "u"
```

  Or alternatively, create a string and use the built-in function `in` such as

```
i not in "aieou"
```

Solution compiled by Russell Saerang.