CS-UY 1134 - Spring 2018 Midterm 1

Kiersten Page

TOTAL POINTS

64 / 100

QUESTION 1

1 Question 1 12 / 18

✓ - 6 pts 2 answers wrong, Correct: FTTFFT

QUESTION 2

2 Question 2 12 / 12

✓ - 0 pts Correct

QUESTION 3

3 Question 3 5 / 8

✓ - 3 pts incorrect number of iterations of loop

QUESTION 4

4 Question 4 1 / 15

✓ - 5 pts Part1: incorrect runtime

✓ - 5 pts Part 2: incorrect runtime

✓ - 4 pts Part 3: incorrect runtime

QUESTION 5

5 Question 5 6 / 12

✓ - 2 pts Incorrect/ No local runtime cost in recursion tree

✓ - 2 pts Incorrect runtime

✓ - 2 pts Incorrect calculations

QUESTION 6

6 Question 6 19 / 20

✓ - 1 pts Inaccurate

QUESTION 7

7 Question 7 9 / 15

✓ - 4 pts does not add value returned by recursive call to sum computed locally

✓ - 2 pts no return value in recursive case

Name: Kiersten Page Net ID: kcp394

Question 1 (18 points)
For each of the following, state if it is true or false (circle your choice):

I. $5n^2 + 3n + 2 = \theta(n)$ — TRUE / (FALSE)

II. $5n^2 + 3n + 2 = \theta(n^2)$ — (TRUE) / FALSE

III. $\log_2(n) = O(n)$ — TRUE / (FALSE)

IV. $\log_2(n) = \theta(n)$ — TRUE / (FALSE)

V. $1 + 2 + 3 + \cdots + n = O(n)$ — (TRUE) / FALSE

VI. $1 + 2 + 3 + \cdots + n = \Omega(n)$ — (TRUE) / FALSE

Question 2 (12 points)

What is printed when the following Python code is executed?

```
import copy
lst1 = [['a','b'], 'c', ['d','e']]
lst2 = lst1
lst3 = copy.copy(lst1)
lst4 = copy.deepcopy(lst1)

lst1[0][1] = 'x'
lst1[1] = 'y'
lst1[2].append('z')
print("lst1 =", lst1)
print("lst2 =", lst2)
print("lst3 =", lst3)
print("lst4 =", lst4)
```

lst1 = [['a', x], y, ['d', 'e', z]]

c won't change

original

Output:

lst1 = [[a, x], y, [d, e, z]]

lst2 = [[a, x], y, [d, e, z]]

lst3 = [[a, x], c, [d, e, z]]

~~lst4 = [['a', 'b'], 'c', ['d', 'e']]~~

lst4 = [[a, b], c, [d, e]]

Name: Kiersten Page Net ID: Kcp 394

Question 3 (8 points):
What is printed when the following Python code is executed?

```
def gen(n):
    curr = 0
    sum = 0
    while curr < n:
        curr = curr + 1
        sum += curr
        yield curr
    yield sum

for num in gen(3):
    print( num, end = ' ')
print([i*i for i in gen(3)])
```

gen(3)

curr = 1 sum = 1 yield 1
curr = 2 sum = 3 yield 2
curr = 3 stop
yield

Output:

1, end =
2, end =
[4, 4, 3, end =
[1, 4, 9]

Name: Kirsten Fuyo Net ID: kef394

Question 4 (15 points)

1. Given the following definition:

```
def func1(n):
    sum = 0
    i = 1
    while (i <= n*n):
        sum += i
        i += 2
    return sum
```

What is the worst case running time of `func1(n)`? Give a short explanation of your answer.

$T(n) = \theta(\ n\ )$

As i increments by 2, the total runtime is equal to $2n-1$, which is a runtime of $\theta(n)$

2. Given the following definition:

```
def func2(n):
    sum = 0
    i = 1
    while (i <= n):
        sum += i
        i *= 2
    return sum
```

What is the worst case running time of `func2(n)`? Give a short explanation of your answer.

$T(n) = \theta(\ n\ )$

Due to i being multiplied by 2 within the while loop, the worst case runtime would be $\theta(n)$. $(1+2+4+8+\ldots+n) = \theta(n)$

Name: Kiersten Page Net ID: kcp 394

3. Given the following definition:

```
def func3(lst):
    n = len(lst)  O(1)
    i = 1   O(1)
    res = []  O(1)
    while (i <= n):
        curr = lst[ : i]  O(i)
        res.append(curr)  O(1)
        i *= 2
    return res
```

20 8

i=1
i=2
i=4
i=8
i=16

only 4 iterations

If `lst` is a list of n integers, what is the worst case running time of `func3(lst)`? Give a short explanation of your answer.

$T(n) = \theta($ $\log_2(n)$ $)$

As the while loop runs and i is multiplied by 2 each time and as append resizes the list but runs in O(1), the entire function will run in $\theta(\log_2(n))$

* $1+2+4+8 \ldots \log_2 n = \theta(\log_2(n))$

Name: Kirsten Page Net ID: Kep394

Question 5 (12 points)
You are given the following recursive function:

```
def mystery(s):
    if(len(s) <= 1):
        return s
    else:
        rest_res = mystery(s[1 : ])
        return rest_res + s[0]
```

s[0] = a

1 to end

1) What would be returned by the call `mystery('abc')`:

Cba

2) Give a meaningful name to `mystery` (A name that describes what this function computes).

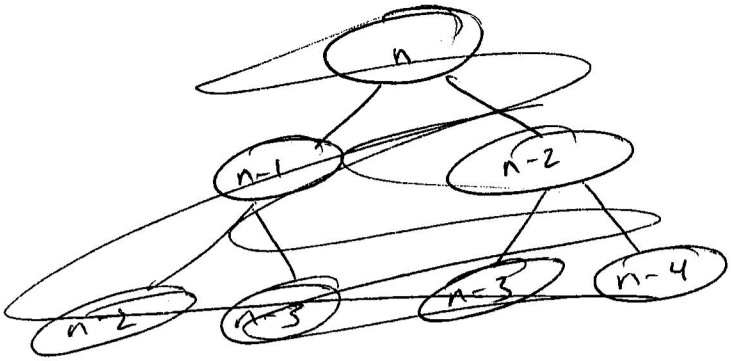
reverse_string

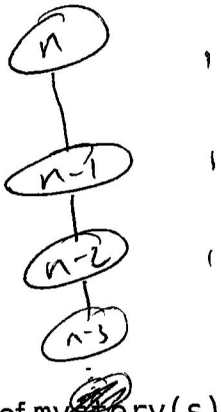
3) Assuming s is a string with n characters, analyze the worst case running time of the call `mystery(s)`:
   a) Draw the recursion tree (showing the recursive calls hierarchy, and their local cost).
   b) Conclude the total (asymptotic) running time of the function.

Note: Write your answers in the next page.

Name: Kiersten Page Net ID: kcp394

i. Draw the **recursion tree** for `mystery(s)`





$\Theta(n)$

ii. **Conclude the running time** of `mystery(s)`

$T(n) = \theta($ n $)$

Calculations:

the call is only called recursively once through each cycle, and the rest_rest decreases by 1 each time, the total running time will be $\Theta(n)$.

Name: Kiersten Page Net ID: Kep 394

Question 6 (20 points)

We define the *palindrome-version* of a string s, to be a string with all the characters of s, followed by the same characters, in a reverse order.
For example, the *palindrome-version* of *"abc"* is *"abccba"*. abccba

Implement the function:

```
def make_palindrome_version(s)
```

This function gets a string, `s`. When called, it should return the palindrome-version of `s`.

For example, the call `make_palindrome_version('abc')`, should return `'abccba'`.

Implementation requirements:

Your function should run in **worst case linear time**. That is, if there are n characters in `s`, calling `make_palindrome_version(s)` will run in $\theta(n)$.

Hint:

You may want to use the `join` string method, which is guaranteed to run in linear time.
As a reminder, the `join(iterable)` method, is a string method, that returns a string, which is the concatenation of the strings in the `iterable` iterable-collection. The separator between elements, is the string providing (calling) this method.
A `TypeError` is raised if there are any non-string values in `iterable`.

For example, the call `';'.join(['abc', 'def', '123'])` would return `'abc;def;123'`
We don't know how the `join` method is implemented, however it is guaranteed that its running time is linear. That the running time of the `join` method is $\theta(n)$ where n is the length of string that `join` returns.

Note: Write your implementation on the next page.

Name: Kiersten Page Net ID: kep394

```
def make_palindrome_version(s):
    copy_string = []
    reverse_string = []
    for character in s:
        copy_string.append(character)
    for i in range(0, len(copy_string), -1):
        reverse_string.append(copy_string[i])
    s2 = "".join(reverse_string)
    return (s+s2)
```

Name: Kiersten Page Net ID: kep394

Question 7 (15 points)

Give a **recursive** implementation for the function:

```
def sum_of_powers_of_2(n)
```

This function is given, a positive integer, n. When called, the function should return the sum of the first n powers of 2, plus the sum of their multiplicative inverses.

For example:

- the call `sum_of_powers_of_2(3)`, will return 14.875,
  Since: $\frac{1}{8}+\frac{1}{4}+\frac{1}{2}+2+4+8 = 14.875$,
- the call `sum_of_powers_of_2(4)`, will return 30.9375,
  Since: $\frac{1}{16}+\frac{1}{8}+\frac{1}{4}+\frac{1}{2}+2+4+8+16 = 30.9375$,

Implementation requirements:

- **Your function must be recursive**
- In this question, don't mind about the running time of your function.

Note: Write your implementation on the next page.

```
i = n
while (i > 0):
    if i == n:
        sum += ((1/(2**i)) + (2**i))
        i -= 1
    else:
        if i == 1:
            sum += ((1/2) + 2)
        else:
```

Name: Kiersten Page Net ID: kep344

```
def sum_of_powers_of_2(n):
    if n==0:
        return sum
    else:
        sum += ((1/(2**n)) + (2**n))
        sum_of_powers_of_2(n-1)
```

Name: Preston Reyes Net ID: kep394

EXTRA PAGE IF NEEDED

Note question numbers of any questions or part of questions that you are answering here.
Also, write "ANSWER IS ON LAST PAGE" near the space provided for the answer.