Data Structures - Fall 2019
Exam 1

s0_name_and_id.py

Open the file and make sure you type your last name, first name, and UTEP ID

s1_recursion.py

Problem 1 (10 points): Provide an implementation for the *count_even* method. This method receives a non-negative int n and returns the number of digits in n that are even (zero is an even number) - Use recursion - no loops.

Example1: count_even(285) -> 2
Example2: count_even (565891) -> 2
Example3: count_even (2468) -> 4
Example4: count_even (1357) -> 0
Example5: count_even (130570) -> 2

s2_iterative_time_complexity.py

Problem 2 (4 points): What is the running time (big-O) of the following function? For all problems, assume that *a* is a list.

```
def f(a):
    for i in range(0, len(a), len(a)):
        print(a[0])
```

Option 0	**Option 1**	**Option 2**	**Option 3**	**Option 4**	**Option 5**
O(1)	O(n)	O(log(n))	O(n^2))	O(n log(n))	O(n^3)

Problem 3 (4 points): What is the running time (big-O) of the following function?

```
def f(a):
    for i in range(0, len(a) * len(a), len(a)):
        print(a[0])
```

Option 0	Option 1	Option 2	Option 3	Option 4	Option 5
O(1)	O(n)	O(log(n))	O(n^2))	O(n log(n))	O(n^3)

Problem 4 (4 points): What is the running time (big-O) of the following function?

```
def f(a):
    for i in range(len(a)):
        for j in range(i):
            print(a[0])
```

Option 0	Option 1	Option 2	Option 3	Option 4	Option 5
O(1)	O(n)	O(log(n))	O(n^2))	O(n log(n))	O(n^3)

Problem 5 (4 points): What is the running time (big-O) of the following function?

```
def f(a):
    for k in range(len(a)):
        for j in range(len(a) // 2):
           i = len(a)

           while i > 0:
               print(a[0])
               i = i // 2
```

Option 0	Option 1	Option 2	Option 3	Option 4	Option 5
O(n^3)	O(n^2)	O(n^2 log n)	O(n^3log n)	O(n log(n))	O(n)

s3_recursive_time_complexity.py

Problem 6 (4 points): What is the recurrence equation that describes the running time of the following recursive function? T(n) = a T(n / b) + n ^ k. What are the values of a, b, and k?

```
def f(a, n): # First call: f(a, len(a))
   if n > 0:
       f(a, n // 4)
       for i in range(len(a)):
           print(a[0])
```

Problem 7 (4 points): What is the recurrence equation that describes the running time of the following recursive function? T(n) = a T(n / b) + n ^ k. What are the values of a, b, and k?

```
def f(a, n): # First call: f(a, len(a))
   if n > 0:
       for i in range(2):
           for j in range(3):
               f(a, n // 2)
               f(a, n // 2)

       print(a[0])
```

Problem 8 (4 points): What is the recurrence equation that describes the running time of the following recursive function? T(n) = a T(n / b) + n ^ k. What are the values of a, b, and k?

```
def f(a, n): # First call: f(a, len(a))
   if n > 0:

       f(a, n // 8)
       f(a, n // 8)

       for i in range(10 // 2):
           for j in range(80 // 4):
               for k in range(len(a) // 8):
                   print(a[0])
```

Master Theorem:

T(n) = O($n^{log_b a}$) if a>b^k

T(n) = O(n^klog n) if a=b^k

T(n) = O(n^k) if a<b^k

Problem 9 (4 points): Solve the following recurrence equation:

T(1) = 1
T(n) = 2 T(n / 2) + n

Option 0	**Option 1**	**Option 2**	**Option 3**	**Option 4**	**Option 5**
O(n^3)	O(n^2)	O(n^2 log n)	O(n^3log n)	O(n log(n))	O(n)

Problem 10 (4 points): Solve the following recurrence equation:

T(1) = 1
T(n) = 2 T(n / 2) + 1

Option 0	Option 1	Option 2	Option 3	Option 4	Option 5
O(n^3)	O(n^2)	O(n^2 log n)	O(n^3log n)	O(n log(n))	O(n)

Problem 11 (4 points): Solve the following recurrence equation:
T(1) = 1
T(n) = 6 T(n / 2) + n^3

Option 0	Option 1	Option 2	Option 3	Option 4	Option 5
O(n^3)	O(n^2)	O(n^2 log n)	O(n^3log n)	O(n log(n))	O(n)

Problem 12 (4 points): Consider the following recurrence equation: T(n) = 2 T (n / 2) + n
Solve the equation by iteration.
T(1) = 1
T(2) = ?
T(4) = ?
T(8) = ?
T(16) = ?
T(32) = ?

What is T(32) ? Your answer must be an integer

s4_activation_records.py

To answer the following three questions, trace the execution of p1(3,4,2) using activation records. Every time you create an activation record, assign it an ID starting from 0.

```
def p1(n, x, y):

    if n > 1:
        p1(n - 2, y - 1, x - 1)
        print("n = ", n, "x = ", x, "y =", y)
        p1(n - 1, x + 1, y + 1)
    else:
        print("n = ", n, "x = ", x, "y =", y)
```

Problem 13 (5 points): How many activation records did you create? The initial call to p1 counts as the first activation record. If you drew 3 boxes, the answer should be 3. If you drew 5 boxes, the answer should be 5, etc.

Problem 14 (5 points): What are the values of n, x, and y in activation record #2? Activation record #0 is the one you created for p1(3,4,2).

Problem 15 (5 points): What is the LAST line printed on the console?

Option 0	Option 1	Option 2	Option 3	Option 4	Option 5
n = 0 x = 6 y = 4	n = 1 x = 6 y = 4	n = 2 x = 5 y = 3	n = 3 x = 4 y = 2	n = 1 x = 4 y = 6	n = 1 x = 5 y = 5

s5_lists_1.py

To answer the following three questions, trace the following piece of code:

```
class Node:
   def __init__(self, item, next):
       self.item = item
       self.next = next

x = None
y = None
for i in range(3):
    x = Node(i + 1, y)
    y = Node(i * 2, x)

print(y.next.item) # Print statement 0
x.next.item -= 2
y.next.next.item += 4
print(y.next.next.item) # Print statement 1
x = y.next
y = x.next.next
print(y.item) # Print statement 2
```

Problem 16 (5 points): What integer does 'Print statement 0' print to the console?

Problem 17 (5 points): What integer does 'Print statement 1' print to the console?

Problem 18 (5 points): What integer does 'Print statement 2' print to the console?

s6_lists_2.py

Problem 19 (7 points): Complete the implementation of the method *remove_last* <- Method removes the last node in the list if it exists.

Problem 20 (7 points): Complete the implementation of the method *contains* <- Method that receives an item as input and returns True if there is a node in the list that stores that. If item is not in the list, the method should return False

Problem 21 (7 points): Complete the implementation of the method *get* <- Method that returns the item associated with the node at position *index* in the list. Return None if the index is invalid.

Problem 22 (7 points): Complete the implementation of the method *add* <- Method that receives an item as input, creates a node that stores this value, and adds it to the list at the specified location (index). If the index is invalid, your method should not do anything.

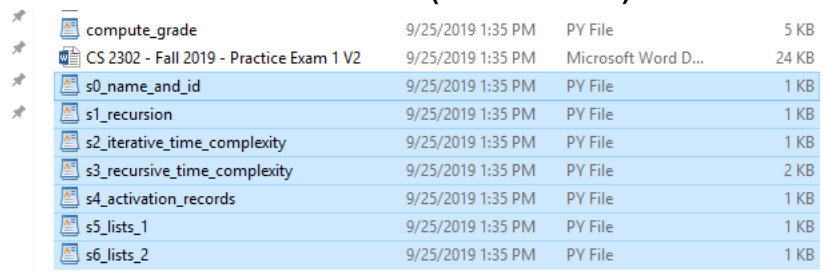
Total number of points: 112 points, graded out of 100 points

----- HOW TO UPLOAD YOUR EXAM-----

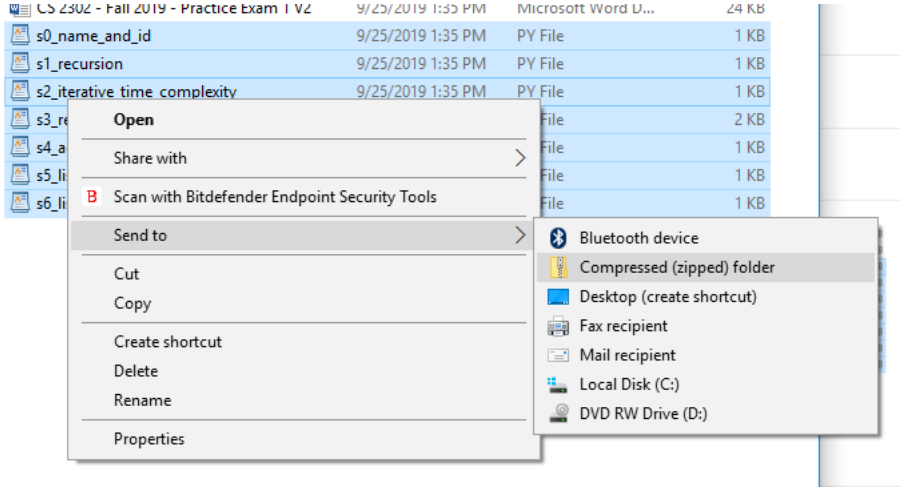
Make sure compute_grade runs! If you have an infinite loop or if the code fails to compute your grade, you will automatically get a 0

Windows 10:

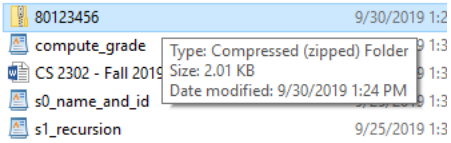
1. **Select the 7 section files (from s0 to s6)**



2. **Right click on any of the selected files and do "Send to -> Compressed (zipped) Folder"**



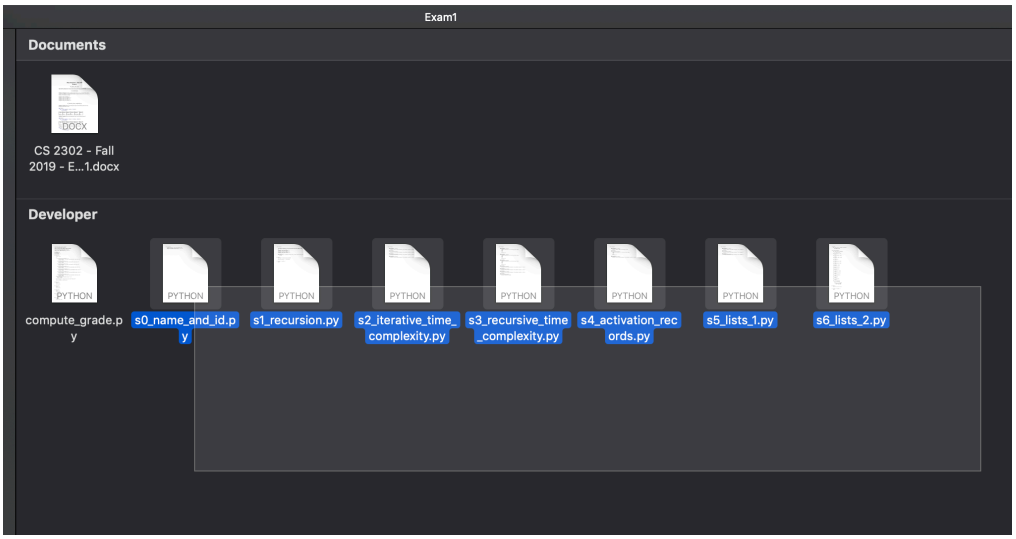
3. **A zip file will be created. Use your UTEP ID to rename this file. The final name must be: <UTEP_ID>.zip**



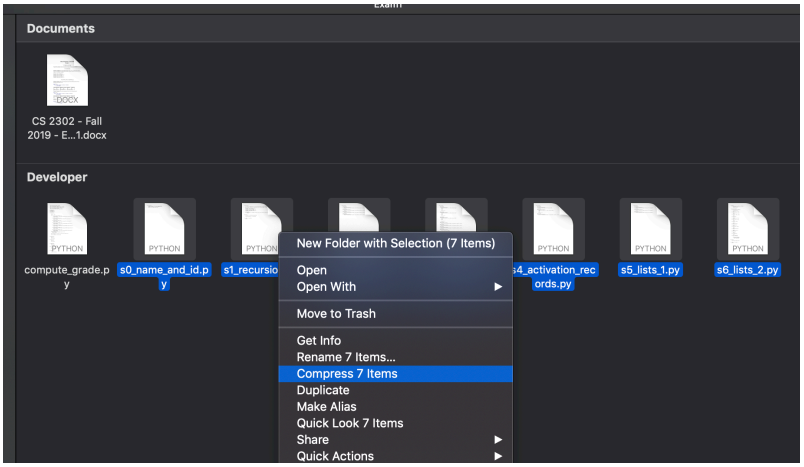
4. **Upload the zip file (Blackboard).**

macOS:

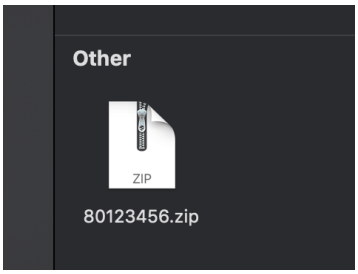
1. **Select the 7 section files (from s0 to s6)**



2. **Right click on any of the selected files and click “Compress 7 items”**



3. **An “Archive.zip” file will be created. Use your UTEP ID to rename this file. The final name must be: <UTEP_ID>.zip**



4. **Upload the zip file (Blackboard)**