Data Structures - Fall 2019
Make Up 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 *change_x_y* method. Given a string, compute recursively (no loops) a new string where all the lowercase 'x' chars have been changed to 'y' chars.

Example1: change_x_y(“codex”) -> "codey"
Example2: change_x_y (“xxhixx”) -> " yyhiyy "
Example3: change_x_y (“xhixhix”) -> "yhiyhiy"

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(len(a), len(a), 1):
       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(len(a), 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, 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 5 (4 points): What is the running time (big-O) of the following function?

```
def f(a):
    for k in range(len(a) * 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)
       f(a, n // 4)
       for i in range(0, len(a), 2):
           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(4):
           for j in range(2):
               f(a, n // 2)

       for i in range(4):
           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 // 3)

       for i in range(len(a) // 2):
           for j in range(len(a) // 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) = 8 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 10 (4 points): Solve the following recurrence equation:

T(1) = 1

T(n) = 7 T(n / 4) + n^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)

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

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

Problem 12 (4 points): Consider the following recurrence equation: T(n) = 3 T (n - 1) + 1
Solve the equation by iteration.
T(1) = 1
T(2) = ?
T(3) = ?
T(4) = ?

What is T(4) ? Your answer must be an integer

s4_activation_records.py

To answer the following three questions, trace the execution of p1(4,2,1) using activation records. Every time you create an activation record, assign it an ID starting from 0.

```
def p1(n, x, y):
    if n > 2:
        p1(n-1, y, x)
        print("n = ", n, "x = ", x, "y = ", y)
        p1(n - 2, 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 #3? Activation record #0 is the one you created for p1(4,2,1).

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

s5_lists_1.py

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

```
x = None
y = None

for i in range(2, 5):
    y = Node(i, x)
    x = Node(i - 1, y)

print(x.next.next.item)  # Print Statement 0
y.next = x.next.next.next

print(y.next.next.item)  # Print Statement 1

x.next.next.next.item += x.item + y.item  # Notice the +=

print(x.next.next.next.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_first* <- Method that removes the first node in the list if it exists.

Problem 20 (7 points): Complete the implementation of the method *has_duplicates* <- Method that returns True if and only if the list contains any duplicate items. If the list does not contain any duplicates, this method returns False.

Problem 21 (7 points): Complete the implementation of the method *clear* <- Method that clears the list. That is, it removes all the nodes in the list.

Problem 22 (7 points): Complete the implementation of the method *remove* <- Method that receives an index as input and removes the node located at that index. If the index is invalid, your method should not do anything.

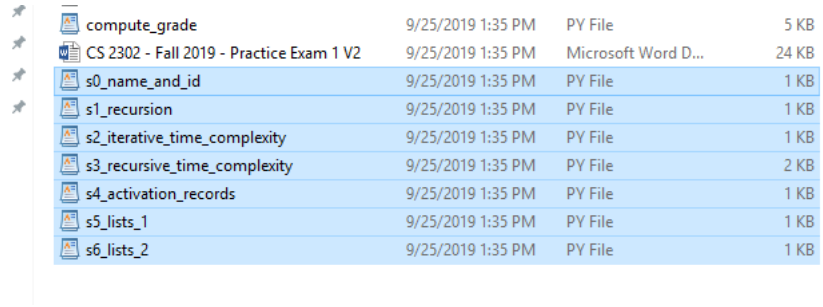
Total number of points: 112 points, graded out of 100 points

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

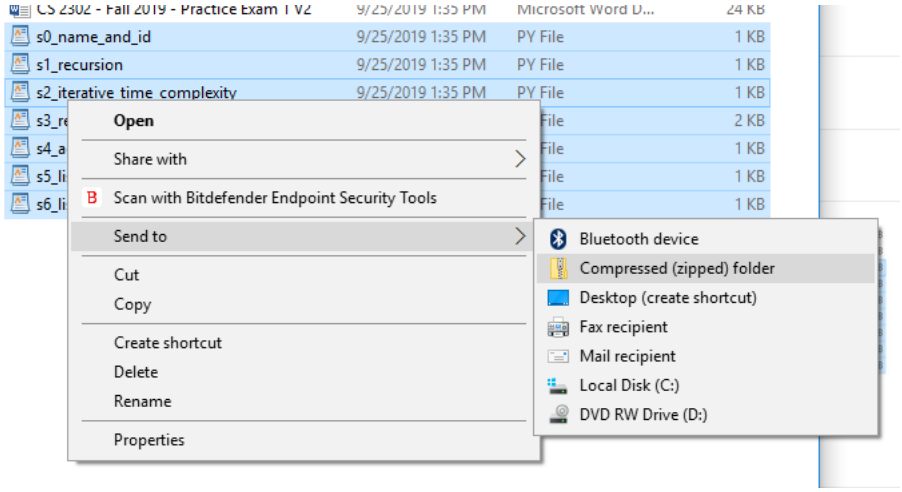
Make sure compute_grade runs! If you have an infinite loop or if the code fails to compute your grade, you will automatically get a 0

Windows 10:

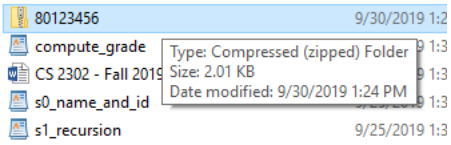
1. **Select the 7 section files (from s0 to s6)**



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



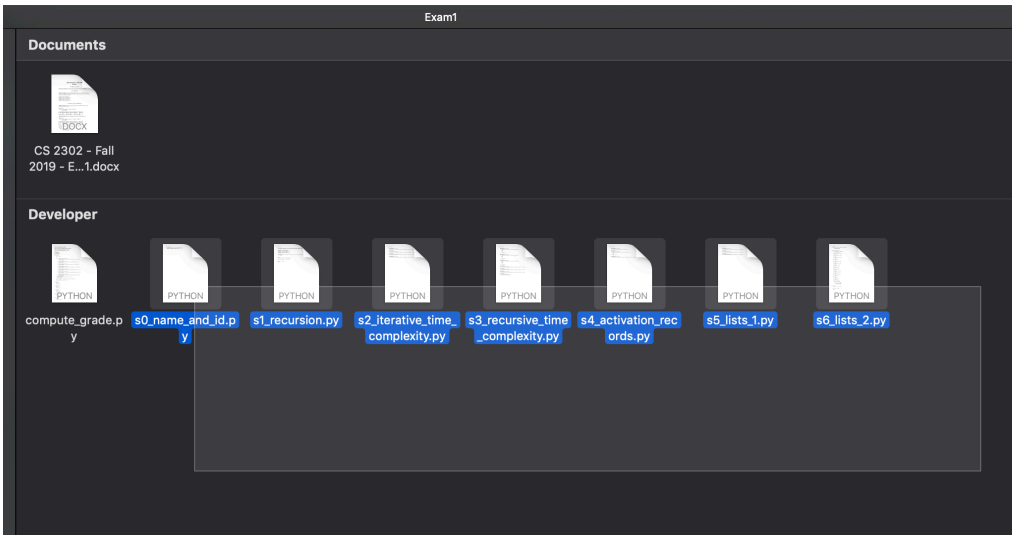
3. **A zip file will be created. Use your UTEP ID to rename this file. The final name must be: <UTEP_ID>.zip**



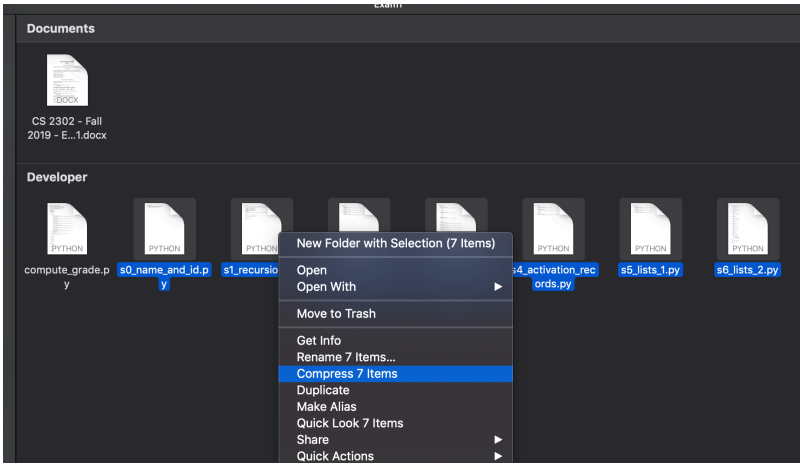
4. **Upload the zip file (Blackboard).**

macOS:

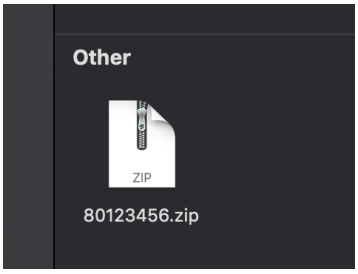
1. **Select the 7 section files (from s0 to s6)**



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)**