

15-112
Fall 2024 Exam 2
November 12, 2024

Name:

Andrew ID:

- You may not use any books, notes, or electronic devices during this exam.
- You may not ask questions about the exam except for language clarifications.
- Show your work on the exam to receive credit.
- Write your answers in the specified places. If you run out of space for an answer, you may write on the backs of pages, but make sure to write a note telling the grader where to look for the rest of your answer.
- All code samples run without crashing. Assume any imports are already included as required.
- You may assume that math, string, and copy are imported; do not import any other modules.

Don't write anything in the table below.

Question	Points	Score
1	20	
2	16	
3	20	
4	16	
5	12	
6	16	
Total:	100	

1. **Code Tracing:** Indicate what the following programs print. Place your answers (and nothing else) in the boxes below the code.

(a) (5 points) CT1

```
def ct1(L):
    for e in L[:]:
        if e % 2 == 0:
            L[e] += 1
        elif e in L:
            L.remove(e)
    print(L)
    return sum(L[1::2])

print(ct1([8, 0, 6, 7, 7, 6, 6, 9, 0, 9]))
```

(b) (5 points) CT2

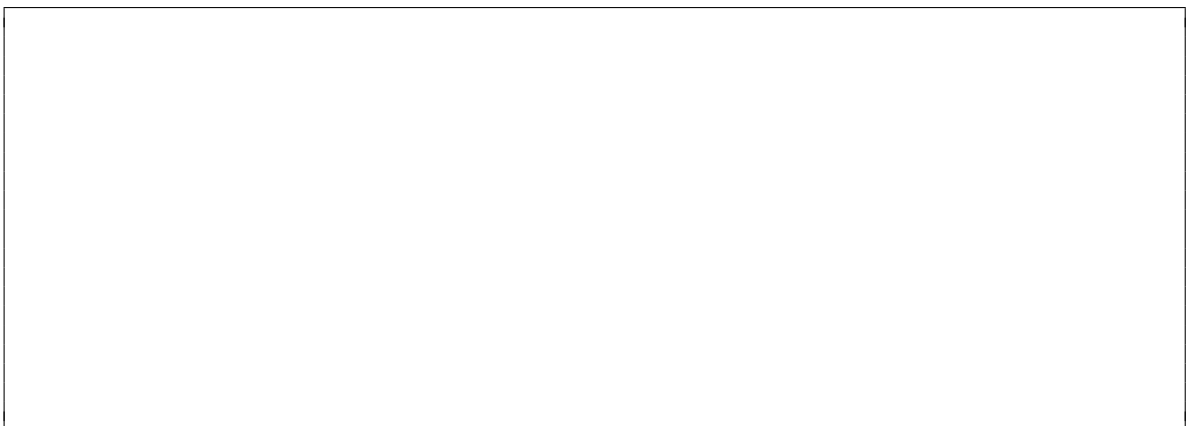
```
def ct2(L):
    d = dict()
    for i in range(len(L)):
        d[L[i]] = d.get(L[i], 0) + i
    print(d)
    L = sorted([a[1] for a in d.items()])
    print(L)

print(ct2([7, 4, 3, 7, 4, 7, 2]))
```

(c) (5 points) CT3

```
def ct3(b, a):
    a = b
    b = copy.copy(a)
    c = copy.deepcopy(a)
    c[0][0] = "Joe"
    a[0] = a[0] + ["Asad"]
    a[0] += [8]
    c[0] = b[0]
    b[1][2] = "Bye"
    a += [112]
    print(f"a: {a}")
    print(f"b: {b}")
    print(f"c: {c}")

a = [[1, 3], [4, "Hi", 5]]
b = [[1, 2, 3], [4, 5, 6]]
ct3(a, b)
print(f"a: {a}")
print(f"b: {b}")
```

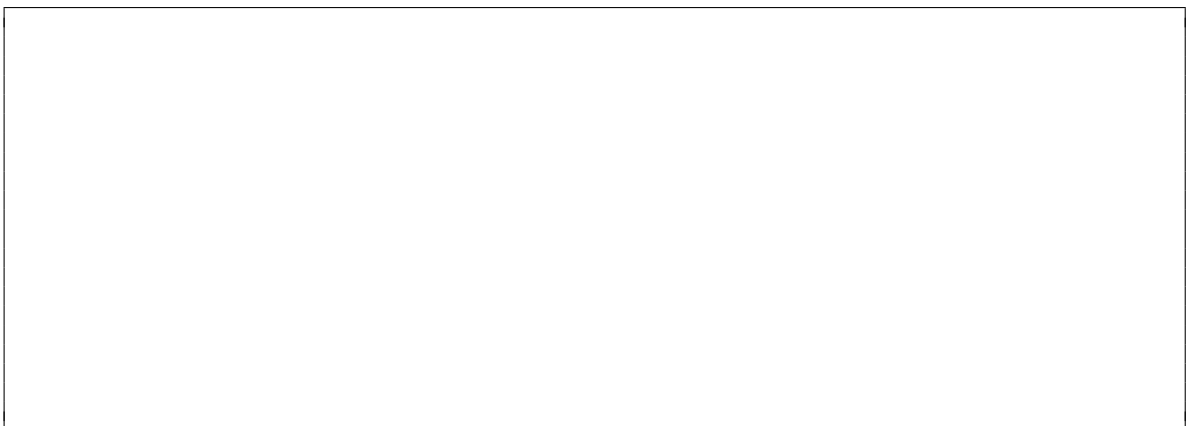


(d) (5 points) CT4

```
def helper(s):
    if len(s) == 0:
        return 0
    return int(s[0]) + helper(s[1:])

def ct4(L):
    if len(L) == 1:
        return helper(str(L[0]))
    elif len(L) == 0:
        return 0
    p1 = ct4(L[:len(L)//2])
    p2 = ct4(L[len(L)//2:])
    res = max(p1, p2)
    print(f"{L}: {res}")
    return res

print(ct4([13, 18, 14, 10, 26]))
```



2. (16 points) **Short Answer:** For each of the four functions shown below, write the total Big-O runtime of the function in terms of N in the box to the right of the code.

```
def sa1(lst): # lst is a list of length N
    a = lst[:len(lst)//10]
    for elem in a:
        if elem % 2 == 1:
            return False
    return True
```

```
# s1 and s2 are strings of length N
def sa2(s1, s2):
    v = []
    for c in s1:
        if s2.count(c) == s1.count(c):
            v.append(s2.count(c))
        else:
            v.append(-1)
    t = 0
    for e in v:
        if e > 0:
            t += 1
        else:
            t -= 1
    return t
```

```
# a is a set and b is a list. Both are length N.
def sa3(a, b):
    for item in a:
        if item in b:
            return False
    return True
```

```
# a is a list and b is a set. Both are length N.
def sa4(a, b):
    for item in a:
        if item in b:
            return False
    return True
```

3. (20 points) **Free Response:** Nearest to 112

Write the non-destructive function `kNearest(L, k)` which, given a list of points in multidimensional space `L` (each point is a list of coordinates) and an integer `k` returns the `k` points nearest to the point `(1.12, 1.12, ...)`. The returned list should be ordered from closest to furthest.

Consider the following test cases:

```
assert kNearest([(2, 5), (1, 2), (3, 4), (1, -1), (-5, 7)], 3) == [(1, 2), (1, -1), (3, 4)]
assert kNearest([(4, 5, 6), (7, 3, 6), (3, 7, 4), (5, 4, 3)], 2) == [(5, 4, 3), (3, 7, 4)]
```

For full credit, your solution should be **better than** $O(N^2)$, where N is the length of `L`. For partial credit (-5 pts), it may be $O(N^2)$.

Notes:

1. The formula for 2-dimensional distance is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. For 3-dimensions it is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$. This pattern continues for higher dimensions:
 $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 + \dots}$
2. If multiple points have the same distance and this impacts the return, then you may break the tie however you would like.

Write the efficiency of your solution in this box:

4. (16 points) **Free Response:** Letter Adjacency Map

Write the function `letterAdjMap(s)` which, given a string `s` returns a dictionary that maps letters from `s` to a map of the count of letters that immediately follow each letter.

For example, in the string `"Hey, that's Henry!"` the letter `h` is followed by `e` twice (once in `"Hey"` and once in `"Henry"`) and `a` once (in `"that"`). The returned dictionary would contain `{"h": {"e": 2, "a": 1}}` in addition to the data for other letters.

Consider the following test case:

```
assert letterAdjMap("Hey, that's Henry!") == {
    "h": {"e": 2, "a": 1},
    "e": {"y": 1, "n": 1},
    "t": {"h": 1},
    "a": {"t": 1},
    "n": {"r": 1},
    "r": {"y": 1},
}
```

Notes:

1. When doing the analysis, case does not matter. (So `'h'` and `'H'` are the same character.)
2. In the returned dictionary, all letters should be lowercase.
3. Only letters are considered. Other characters, such as spaces and punctuation, are not considered.

5. (12 points) **Free Response:** Summing Nested Lists

Your solution to this problem must be entirely recursive. No loops or iterative functions are allowed; their use will result in a score of zero for this problem.

Write the function `sumNestedLists(L)` which, given a list `L` whose elements could be integers or other lists of integers (and so on to any depth) returns the sum of all of the integers in the lists.

Consider the following test cases:

```
assert sumNestedLists([7, [1, 2], 3]) == 13
assert sumNestedLists([1, [2, [3, [4]]], 5]) == 15
```


6. (16 points) **Free Response: OOP**

Write the classes `Light` and `DimmableLight` so that the following test code runs without errors. Do not hardcode against the values used in the testcases, though you can assume the testcases cover the needed functionality. You must use proper object-oriented design or you will lose points.

```
a = Light(500, "Bedroom")
assert str(a) == "Light (lumens=500, location=Bedroom, active=off)"
assert a.isOn() == False
a.toggleSwitch()
assert str(a) == "Light (lumens=500, location=Bedroom, active=on)"
assert a.isOn() == True
a.toggleSwitch()
assert str(a) == "Light (lumens=500, location=Bedroom, active=off)"
assert a.isOn() == False

# The only attributes that matter for equality are the lumens and location
b = Light(500, "Bedroom")
b.toggleSwitch()
assert b.isOn() == True
assert a == b
assert a != Light(501, "Bedroom")
assert a != Light(500, "Bathroom")
assert a != "Light (lumens=500, location=Bedroom, active=off)"

d = DimmableLight(300, "Dining Room")
assert str(d) == "DimmableLight (lumens=300, location=Dining Room, active=off, intensity=100%)"
d.dim(75)
assert str(d) == "DimmableLight (lumens=300, location=Dining Room, active=off, intensity=75%)"
# Invalid dim values do nothing
d.dim(101)
d.dim(-1)
assert str(d) == "DimmableLight (lumens=300, location=Dining Room, active=off, intensity=75%)"
d.toggleSwitch()
assert str(d) == "DimmableLight (lumens=300, location=Dining Room, active=on, intensity=75%)"

# Like a normal light, DimmableLights are equal based only on the lumens and location
e = DimmableLight(300, "Dining Room")
e.toggleSwitch()
e.dim(15)
assert d == e
assert d != DimmableLight(301, "Dining Room")
assert d != DimmableLight(300, "Living Room")
assert d != "DimmableLight (lumens=300, location=Dining Room, active=on, intensity=75%)"
# Lights and DimmableLights are never equal
assert Light(300, "Dining Room") != DimmableLight(300, "Dining Room")

# Both work in sets
s = set()
s.add(a)
s.add(d)
s.add(e)
assert len(s) == 2
assert Light(500, "Bedroom") in s
assert DimmableLight(300, "Dining Room") in s
assert Light(500, "Bathroom") not in s

# Various type checking things
assert isinstance(a, Light)
assert not isinstance(a, DimmableLight)
assert isinstance(d, Light)
assert isinstance(d, DimmableLight)
```

Write your solution on the next two pages

