15-112
Spring 2023 Exam 2
March 23, 2023

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.
- You may use the backs of pages as scratch paper.
- All code samples run without crashing except otherwise specified. Assume any imports are already included as required.
- You may assume that random, math, string, basic_graphics, cmu_112_graphics and copy are imported; do not import any other modules.
- If you need extra space, use the back of the pages. Do not tear off any page. Doing so may be considered an academic integrity violation.

Don't write anything in the table below.

Question	Points	Score
1	10	
2	15	
3	10	
4	15	
5	15	
6	15	
7	20	
Total:	100	

1. **Short answers**

   Answer each of the following *very briefly*.

   (a) (1 point) In 15-112, do we care about constant coefficients, so $O(N^2)$ is better than $O(100N^2)$? Answer YES or NO in the box below.

   (b) (1 point) In 15-112, do we care about lesser-order terms, so $O(N^2)$ is better than $O(N^2 + 10000N + 100)$? Answer YES or NO in the box below.

   (c) (2 points) In one sentence, explain why this code will crash:

```
d = {'a':'b', 'c':'d'}
d['a'] = d['b'] + d['c']
```

   (d) (2 points) The following function is meant to remove spaces within a string recursively. Indicate whether the function is correct or not. If not, explain what's wrong with it.

```
def recRemoveSpaces(s):
    if s[0] == ' ':
        return recRemoveSpaces(s[1:])
    else:
        return s[0] + recRemoveSpaces(s[1:])
```

(e) (2 points) Indicate whether the following code runs without error or not. If not, explain what's wrong with it.

```
class Label:
    def __init__(self, label):
        self.label = label
    def __repr__(self):
        s = "Label<" + self.label + ">"
        return s
    def __eq__(self, other):
        return self.label == other.label

a = Label('15-112')
assert(a == "Label<15-112>")
```

(f) (2 points) Indicate whether the following code is a good example of the proper use of inheritance. If not, explain what's wrong with it.

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height
    def getArea(self):
        area = self.width * self.height
        return area

class Square(Rectangle):  # A square is a rectangle with equal width and height
    def __init__(self, side):  # rewrite the constructor to use a single parameter
        super().__init__(side, side)  # call the parent's constructor
    def getArea(self):
        area = self.width * self.height
        return area
```

2. **Code Tracing**

Indicate what each will print. Place your answer (and nothing else) in the box next to or below each block of code.

(a) (5 points) CT1

```
def ct1(L):
    s = set()
    for v in L:
        if isinstance(v, dict):
            for k in v:
                s.add(v[k])
        else:
            for e in v:
                if e in s:
                    s.remove(e)
        print(s)
    return s

print(ct1(['CB', {2:'A', 'B':3, 4:'C'},
          'BC3', [4, 3, 2]]))
```

(b) (5 points) CT2

```
def ct2Helper(n, m):
    if (n%10)%2 == 1:
        print(n, m)
        m = m * 10 + n%10
    if n > 10:
        return ct2Helper(n//10, m)
    else:
        return m

def ct2(n):
    return ct2Helper(n, 0)

print(ct2(15112))
```

(c) (5 points) CT3

```
class A(object):
    def __init__(self):
        self.b = "sparkling"

    def f(self):
        return self.g()

    def g(self):
        return "apple"

    def h(self, s):
        print(f"{self.b} {s}")
        return self.f()

    def __repr__(self):
        return f'A {self.f()}'

class B(A):
    def __init__(self, s):
        super().__init__()
        self.b = s

    def f(self):
        s=self.g()
        return f"pine{s}"

    def g(self):
        return super().g()

def ct3():
    b = B("still")
    print(b.h("water"))
    print(type(b) == A)
    print(isinstance(b, B))
    print(isinstance(b, A))
    return b

print(ct3())
```

3. **Reasoning Over Code**

For each function, find values of the parameters so that the following functions will return `True`. Place your answer (and nothing else) in the box below each code.

(a) (5 points) ROC1: Indicate the values of `d1` and `d2`.

```
def roc1(d1, d2):
    assert(len(d1) == len(d2))
    assert(len(d1) == 2)
    s = set()
    for k1 in d1:
        if d1[k1] not in d2:
            return False
        s.add(k1)
        s.add(d2[d1[k1]])
    return s == {'a', 'b', 'c'}
```

You must indicate two values, `d1` and `d2`.

(b) (5 points) ROC2: Indicate the value of `s` for the function `roc2(s)` return `True`. Note: `rocHelper2(s)` is a helper function, and your answer should refer to `roc2(s)`.

```
def rocHelper2(s):
    if len(s) < 2:
        return s
    return s[1] + s[0] + rocHelper2(s[2:])

def roc2(s):
    res = rocHelper2(s)
    return res == 'cmu112s23'
```

4. (15 points) **Free Response: Recursive Separate Strings and Non Strings**

   Write the function `recStringsAndNonStrings(L)` that returns a tuple of two lists such that the first list contains all elements in L that are strings, and the second list contains all elements in L that are not strings. The elements in both lists must respect their relative order in L.

   For instance,

```
assert(recStringsAndNonStrings(['hi', 42, 3, 'exam']) == (['hi', 'exam'], [42, 3]))
assert(recStringsAndNonStrings(["exam", "2"]) == (['exam', '2'], []))
assert(recStringsAndNonStrings([1, 1, 2]) == ([], [1, 1, 2]))
```

   Your solution must use recursion. If you use any loops or iterative functions, you will receive no points on this problem.

5. (15 points) **Free Response: Find Zero Triplets**

Write the function `findZeroTriplets(L)` that takes as input a list `L` of integers of length `N` and returns a set of all triplets in the list whose sum is equal to 0. For example, if the given list is `[-1, 0, -3, 2, 1]`, you should return `{(-1, 0, 1), (-3, 1, 2)}` (note that triplets are sorted). If there is no valid triplet, you should return the empty set. You may assume that `L` is a list containing only integers. The *naive* solution would use 3 loops to check all triplets of values in `L`.

You should use sets and/or dictionaries to do this faster than $O(N^3)$

You should also specify the Big-O of your solution in the box below:

Additional Space for Answer to Question 5

6. (15 points) **Free Response: wordChange** You are given two lowercase words `w1` and `w2`, where `w2` is formed by rearranging `w1`'s characters and adding new letters.

   Write the function `wordChange(w1, w2)` that returns what new letters are needed to make `w2` from `w1`. The function must return a sorted list of the characters, including duplicates if needed.

   Hint: Think about the frequencies of each letter, how to use them to solve the task, and how to compute them efficiently.

   For full credit, your solution must have an efficiency of O(N) or better, where N is the length of `w2`. You can get half-credit for less efficient solutions.

```
assert(wordChange("time", "timeless") == ['e', 'l', 's', 's'])
assert(wordChange("listen", "silently") == ['l', 'y'])
assert(wordChange("same", "assessment") == ['e', 'n', 's', 's', 's', 't'])
assert(wordChange("range", "rearrange") == ['a', 'e', 'r', 'r'])
```

Additional Space for Answer to Question 6

7. (20 points) **Free Response: Animation** Write an app such that:

   1. When the app starts, a red dot of radius 50 pixels is centered in the canvas.
   2. When the user presses the left arrow, the dot moves 10 pixels to the left.
   3. When the user presses the right arrow, the dot moves 10 pixels to the right.
   4. Four times per second, the dot's radius grows by 20.
   5. When the user presses `p`, the app is paused or unpaused.
   6. When paused, the dot's radius does not grow, but the arrow keys still work.
   7. When paused, if the user presses `s`, the game takes one step, so the radius grows by 20 pixels.
   8. If any part of the dot ever goes beyond the left or right edge of the canvas, whether from an arrow press or from the radius growing, then the dot returns to the center, and its radius is set to 50 pixels.

   Notes:

   - You must assume the canvas is 400x400.
   - Make reasonable assumptions for anything not specified here.
   - We recommend that to save time writing, you abbreviate `canvas`, `event`, and `app`: use `c`, `e` and `a`, respectively.
   - To solve this, you need to write `appStarted`, `keyPressed`, `redrawAll`, and `timerFired`.
   - You do not need to include any imports or the main function.

Answer space for Question 7

Answer space for Question 7