fullName:____________________________andrewID:____________________________ recitationLetter:_____

15-112 F23

Midterm2

You **MUST** stop writing and hand in this **entire** exam when instructed in lecture.

- You may not unstaple any pages.
- Failure to hand in an intact exam will be considered cheating. Discussing the exam with anyone in any way, even briefly, is cheating. (You may discuss it only once the exam has been posted to the course website.)
- You may not use your own scrap paper. If you must use additional scrap paper, raise your hand and we will provide some. You must hand any scrap paper in with your paper exam, and we will not grade it.
- You may not ask questions during the quiz, except for English-language clarifications. (If you must ask an English-language question, please come out to the nearest aisle after you have our attention. We cannot come to you.) If you are unsure how to interpret a problem, take your best guess.
- You may not use any concepts (including builtin functions) which we have not covered in the notes in weeks 1-11 / units 1-8.
- We may test your code using additional test cases. Do not hardcode.
- Assume almostEqual(x, y) and rounded(n) are both supplied for you. You must write all other helper functions you wish to use, unless we specify otherwise.

Sign your initials below to assert that you will not discuss or share information about the exam with anyone in any way, even briefly, until the exam has been posted on the course website.

x______________

Multiple Choice / Short Answers [2pts ea, 8pts total]

MC1. If we wish to find whether one item exists in a very large unsorted list of length N, which of the following options would we expect to be fastest in the worst case?
Select the best answer (fill in one circle).

○ a) Sorting the list with merge sort, then performing a linear search

○ b) Sorting the list with merge sort, then performing a binary search

○ c) Sorting the list with selection sort, then performing a binary search

○ d) Performing a linear search on the unsorted list

MC2. Of the following, which is the most efficient?
Select the best answer (fill in one circle).

○ a) O(log(N))

○ b) O(N*log(N))

○ c) O(N**0.5)

○ d) O(2**N)

○ e) O(2**0.5)

○ f) O(N**2)

MC3. What is the big O of the following function, which takes a list of length N? Note that some parts of the code are intentionally blanked out.

```
def f(L):
    result = []
    for i in range(_______):
        for j in range(_______):
            k = L[i] + L[j]
            result.append(k)
    return result
```

Select the best answer (fill in one circle).

○ a) O(N*log(N))

○ b) O(1)

○ c) O(2**N)

○ d) O(N**2)

○ e) O(N**3)

○ f) O(N**0.5)

○ g) Need more information to be sure

MC4. Which of the following may require Python to visit all N elements in the list data, assuming N = len(data)? Select ALL that apply.

☐ a)

```
x = data[-1]
```

☐ b)

```
for x in data:
    print(x)
```

☐ c)

```
for i in range(len(data)):
    x = data[i]
    print(x)
```

☐ d)

```
x = max(data)
```

☐ e)

```
if x in data:
    print(“Found it”)
```

☐ f) None of the above

CT1: Code Tracing [6pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct1(d):
    s = set()
    for k in d:
        if k in d[k]:
            s.add(max(d[k]) % k)
    return s

print(ct1(
      { 3:  { 3, 6 },
        4:  { 2 },
        5:  { 2, 5, 9 },
        6:  { 1, 6, 7 },
        7:  { 1, 7, 71 },
        12: { 10, 2, 82 }
      }))
```

CT2: Code Tracing [8pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct2(L):
    d = dict()
    for i in range(len(L)):
        for j in range(i+1, len(L)):
            if L[i] == L[j]:
                d[i+j] = (i,j)
    return d

print(ct2([3,2,1,3,2,4]))
```

CT3: Code Tracing [8pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct3(L, x):
    if L == [ ]:
        return L
    else:
        if L[-1] > x:
            rest = ct3(L[:-1], x+2)
            return rest + [L[-1]]
        else:
            return ct3(L[:-1], x)

print(ct3([1, 2, 3, 7, 6, 11], 2))
```

Free Response 1: encodeWords(wordList) [16pts]

Write the function `encodeWords(wordList)` which takes a list of strings (all lowercase letters) and returns a new list of integers, which we will refer to as the encodedList.

The encodedList is the same length as wordList. The integer at each index in encodedList corresponds to the string present at that same index in wordList. Each integer corresponds to exactly one string.

The integer for each string is determined by its position in an alphabetized (i.e. sorted) list of the **unique** strings in wordList.

Consider the following example:

```
wordList = ['durian', 'banana', 'durian', 'apple', 'clementine', 'apple']
```

The unique strings, in alphabetical order, are:

```
['apple', 'banana', 'clementine', 'durian']
```

So, in the encodedList, 'apple' becomes 0, 'banana' becomes 1, 'clementine' becomes 2, and 'durian' becomes 3. The encoded list is thus:

```
encodedList = [3, 1, 3, 0, 2, 0]
```

Note: For full credit, your function must run in O(NlogN) time or better, where N is the length of the wordList.
Hint: You may find sorted() or .sort() useful in this problem. (You may use these builtins, and you do not need to write your own sorting function.)

```
def testEncodeWords():
    assert(encodeWords(['durian', 'banana', 'durian', 'apple', 'clementine', 'apple'])
                    == [ 3,        1,        3,        0,       2,            0])
    assert(encodeWords(['b', 'a','b','c','d','e','d','c','b','a','b'])
                    == [ 1,   0,  1,  2,  3,  4,  3,  2,  1,  0,  1])
    assert(encodeWords(['little', 'axo', 'lotl', 'likes', 'u', 'a', 'lotl'])
                    == [ 3,        1,     4,      2,       5,   0,   4 ])
    assert(encodeWords(['taters']) == [0])
    assert(encodeWords([]) == [])
```

Begin your FR1 answer on the following page

Begin your FR1 answer here

You may continue your FR1 answer here

Free Response 2: Bottle class [18pts]

Write the Bottle class so that the following test code passes. Do not hardcode the test cases, but you may assume that the parameters are always legal (so, for example, .fill() will always take a positive integer). You must use OOP properly. Do not add unneccessary methods.

```python
def testBottleClass():
    #Make a tall bottle
    myBottle = Bottle('tall')
    assert(myBottle.shape == 'tall')

    #All bottles begin empty
    assert(myBottle.getContents() == 'empty')

    #We can fill a bottle with a material. .fill also returns the current contents.
    assert(myBottle.fill(2, 'water') == '2 oz of water')
    assert(myBottle.fill(2, 'water') == '4 oz of water')

    #We may not add a new material to a bottle that is not empty
    assert(myBottle.fill(1, 'jelly') == 'Do not mix!')
    assert(myBottle.getContents() == '4 oz of water')

    #Bottles can be emptied
    assert(myBottle.empty(3) == 'Poured 3 oz')
    #...but the volume cannot be negative, and can't pour more than it has
    assert(myBottle.empty(3) == 'Poured 1 oz')
    assert(myBottle.empty(0) == 'Poured 0 oz')
    assert(myBottle.getContents() == 'empty')
    #Once a bottle is empty, a different material can be added
    assert(myBottle.fill(3, 'soda') == '3 oz of soda')

    #All bottles have their own shape and contents
    yourBottle = Bottle('round')
    assert(yourBottle.shape == 'round')
    assert(yourBottle.fill(7, 'grades') == '7 oz of grades')
    assert(yourBottle.getContents() == '7 oz of grades')
    assert(myBottle.getContents() == '3 oz of soda')
```

Begin your FR2 answer on the following page

Begin your FR2 answer here

You may continue your FR2 answer here

Free Response 3: getMorphList(L) [18pts]

Write the function getMorphList(L) which takes a non-empty list of strings and returns new result list that meets the following criteria (or None, if no solution exists):

1. The result list is the same length as L
2. The result must contain the same elements as L, potentially in a different order
3. Each string is the same length as its neighbors
4. Each string differs from the string before it by exactly one character

According to these rules, `getMorphList(['but','bat','cut', 'bad'])` could return either `['cut','but','bat','bad']` or `['bad','bat','but','cut']`

However, `getMorphList(['bat','tub'])` returns `None`, because L cannot be reordered to meet rule 4 (and we cannot reorder the characters in the strings).

And `getMorphList(['but','bat','cut', 'bad', 'bard'])` returns `None`, because 'bard' cannot be placed anywhere without violating rule 3.

Items that appear multiple times in L must appear the same number of times in the result, so `getMorphList(['pail', 'pail', 'mail'])` can return `['pail', 'mail', 'pail']`, but not `['mail', 'pail', 'mail']`

Note that some inputs may have many valid solutions, but getMorphList(L) only needs to return one of them.

Note: This function must be solved using recursive backtracking!
Non-recursive functions may receive zero points. Recursive functions that do not meet the requirements for backtracking will only be eligible for up to half credit, even if they pass the test cases.

Note: We have not provided traditional test cases for this function, because your algorithm may return a slightly different list for each input that still meets the rules above.

Begin your FR3 answer on the following page

Begin your FR3 answer here

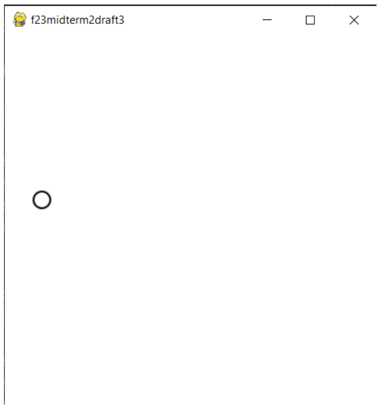
You may continue your FR3 answer here

You may continue your FR3 answer here

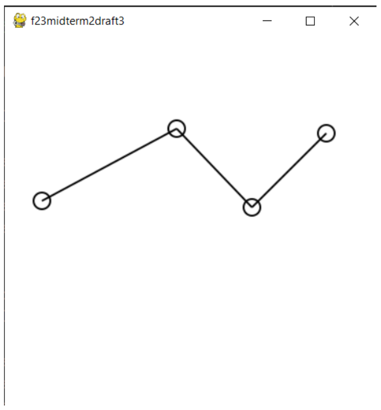
Free Response 4: Line Segment Animation [18pts]

Write an animation with the following features. Read the whole problem before you begin!

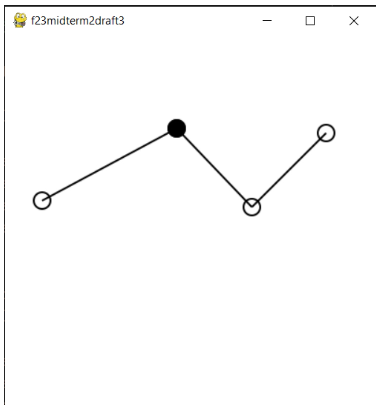
Summary: The app begins with nothing drawn on the canvas. As described below, pressing the mouse will add circles to the canvas. Each circle is connected to the previously-created circle by a line, in the order in which the circles were added to the canvas. (So, the first circle is connected to the second circle, and the second to the third and so on, with the most recent circle connected to the second-most recent circle.) Clicking on a circle selects it, and the 'd' key deletes the selected circle. See below for example screenshots.



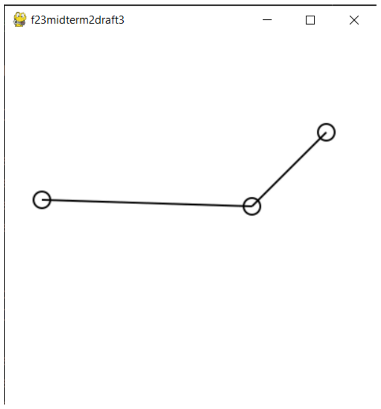
1) Clicking near the left edge has created one circle



2) Clicking three more times gives us 4 circles



3) Clicking the second circle selects it



4) Pressing 'd' deletes that circle

When the mouse is pressed:

- If the location of the mouse press is inside an existing circle:
    - That circle becomes the selected circle.
    - The selected circle's fill is changed from None to black.
    - Only one circle may be selected at any time, so if another circle is already selected, its fill returns to None.
    - If the location is inside more than one circle, only one circle is selected (but it does not matter which)
- If the location of the mouse press is outside of all existing circles:
    - An unselected circle is added to the canvas at that location with a radius of 10, a black border, and no fill.
    - If at least one other circle is already present, a line is drawn connecting the center of the new circle to the center of the previously created circle.
    - If a circle was previously selected, it is no longer selected.

When the 'd' key is pressed:

- If no circle is selected, nothing happens
- If a selected circle exists, that circle is deleted, along with any lines connected to it.
    - If necessary, a new line is drawn so that all circles remain connected in the order in which they were created.
    - When a selected circle is deleted, no new selected circle is chosen unless it is clicked on.

Notes:

- You do not need to use OOP for this problem, nor do we recommend it
- You will be penalized if your code results in an MVC violation
- You may use the default values for anything not specified above, e.g., no need to specify line width or border width for the circles
- Please write your code using the provided function headers for onAppStart, redrawAll, onMousePress, and onKeyPress, . As usual, you may write and use additional helper functions if you wish. (You may also add other event functions, but this is not recommended or necessary.)

Begin your FR4 answer on the following page

Begin your FR4 answer here:

```
from cmu_graphics import *


def onAppStart(app):









def redrawAll(app):
```

Continue your FR4 answer here

```
def onMousePress(app, mouseX, mouseY):
```

Continue your FR4 answer here

```
def onKeyPress(app, key):
















runApp()
```