15-112
Spring 2023 Exam 1
February 16, 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.
- 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.
- Do not use these post-midterm 1 topics/constructs: sets, maps/dictionaries, recursion, or classes/OOP.

Don't write anything in the table below.

Question	Points	Score
1	25	
2	20	
3	10	
4	20	
5	15	
6	10	
Total:	100	

1. **Code Tracing**

(a) (10 points) Write the output for the following short code segments:

Code	Output
`x = 442` `print(x // 10 - x % 10)`	

`def f(a, b):`
`  x = 3*a - b`
`  return x`
`x = 2`
`y = f(1, 12)`
`print(x)`	

`L = [1,5,112]` `print(L[1:])`	
`L = [1,5,112]` `print(L[1::-1])`	

`A = [6, 1, 4]`
`B = A.sort() # it is destructive`
`print(B)`	

`def f(L):`
`  L = L + [112]`
`A = [4, 2]`
`f(A)`
`print(A)`	

`L = [1,5,112]`
`ans = []`
`i = 0`
`for i in range(len(L)):`
`  ans += L[:i]`
`print(ans)`	

`s = "so easy"`
`s = s.split('e')`
`print(s)`	

`s = "yes easy"`
`t = s.replace(' ','\n')`
`print(t)`	

`s="112rocks"`
`i = s.find("1",1)`
`print(s[i:i+3])`	

(b) (5 points) Indicate what the following program prints. Place your answers (and nothing else) in the box next to the code.

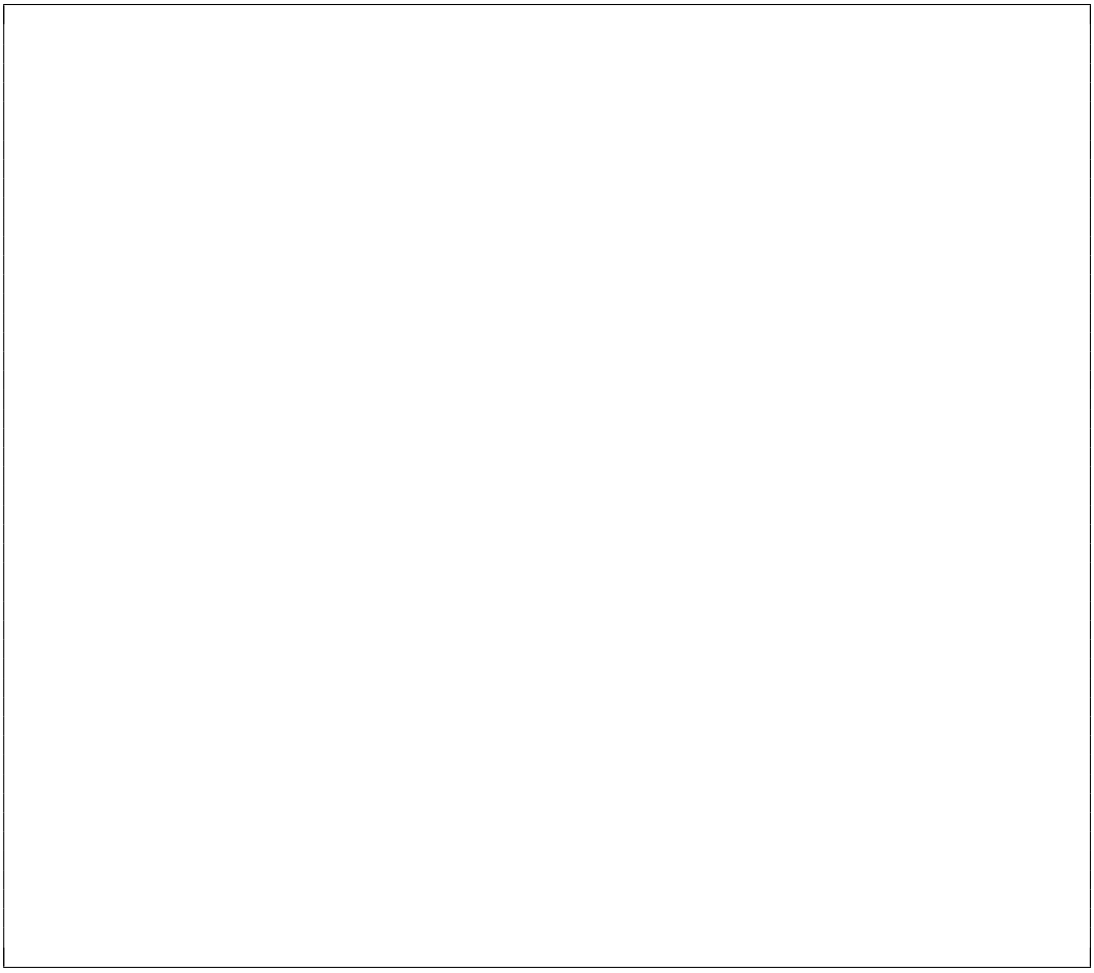
```
def ct1(m):
    x = 1
    while x < 6:
        if x%5 == 0:
            break
        x += 2
        print("x =", x)
    for y in range(m, m+3):
        if y % 3 == 0:
            print("mul3 = ", y)
        elif y % 2 == 0:
            print("even")
        else:
            x += y
    return x

print(ct1(4))
```

(c) (5 points) Indicate what the following program prints. Place your answers (and nothing else) in the box next to the code.

```
def ct2(s):
    res = 0
    t = ""
    u = ""
    for i in range(2, len(s)):
        if s[i:].isdigit():
            res += int(s[i:])
            t += s[i:]
        else:
            u += s[i]
    print("r", res)
    print("t", t)
    print("u", u)
    return s[:2] + u + t

print(ct2('cmu112'))
```

(d) (5 points) Indicate what the following program prints. Place your answers (and nothing else) in the box below the code.

```
def f(M,n):
    print(M)
    s = M[n]
    i = len(s) - 1
    while s[i:].isalpha():
        i -= 1
        M[n] = s[i:]
    print("M[n]=", M[n])
    M = M + [s[:i]]
    return s[:i]

def ct3(L):
    s = ""
    i=0
    while len(s) < 3 and i < 3:
        s += f(L, i)
        i += 1
    print(L, s)

ct3(['12bye', '22hi','123run'])
```



2. **Reasoning Over Code**

(a) (10 points) Choose values for `x` and `y` to cause each of the following expressions to be `True`.

Code	x	y
`len(x) == 4 or x[1:4] == "112"`		N/A
`x//10 == 0 and x % 3 == 1`		N/A
`len(x) == 2 and x[0].lower() == x[1].lower()`		N/A
`len(x + y) == 5 \` `  and (x + y).split('-') == ['ab', 'cd']`		
`len(x) == 2 and x[0] == [0,1] \` `  and x[1] == [x[0][1], x[0][0]]`		N/A

(b) (5 points) Find the argument for the following function to cause it to return `True`. Place your answer (and nothing else) in the box next to the code.
Hint: Try a few numbers.

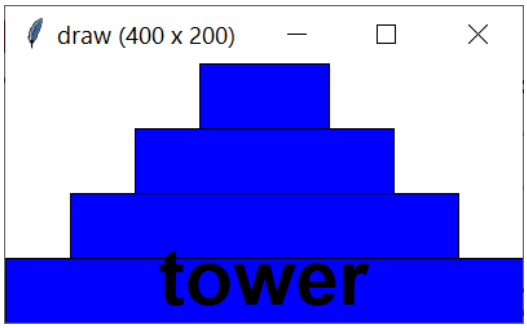
```
def roc1(x):
    assert(10**2 > x > 10**0)
    y = 3
    for z in range(1, x, 2):
        y += z
    return (x % 2 == 1 and y == 19)
```

(c) (5 points) Find the argument for the following function to cause it to return `True`. Place your answer (and nothing else) in the box below the code.

```
def roc2(s):
    t = "a2b1c1"
    assert(isinstance(s, str) and len(s) == len(t))
    length = len(s)
    result = ""
    for index in range(length):
        c1 = s[index]
        c2 = t[length - 1 - index]
        if c1 == c2 :
            result += c2
        elif c1 in t :
            return False
    return result == str(112)
```

3. (10 points) **Fill-in-the-blanks:** Graphics

Fill in the blanks (A,B,C,D,E) so that this code draws the given image (shown below). Do not add any new lines. Just fill in the blanks. The image does not need to resize and you may assume the canvas is 400x200.




```
import basic_graphics

def draw(canvas, width=400, height=200):
    dx = 100 # Width of top-most rectangle
    dy = 50 # Height of top-most rectangle
    for i in range(4):

        x0   = ___________________________ # (A)

        y0   = i*dy


        x1   = ___________________________ # (B)


        y1   = ___________________________ # (C)


        col  = ___________________________ # (D)

        canvas.create_rectangle(x0, y0, x1, y1, fill=col)
    canvas.create_text(width//2, height, text='tower',

                       anchor=_____________,# (E)
                       font='Arial 30 bold')

basic_graphics.run(width=400, height=200)
```

4. **Free Response:** Doubly Numbers

   Do not use dictionaries, sets, try/except, or recursion on this problem. If you do, you will receive a 0.

   Note: For full credit, you may not use strings or lists in your solution. However, you will receive **half credit** *for a fully correct solution that uses strings and/or lists.*

   We'll say that an integer is a *doubly number* (coined term) if it is a positive integer such that...

   - It has an even length
   - If the digits are paired from left to right, the digits within the pairs are all equal: $\underbrace{aa}_{=}\underbrace{bb}_{=}\underbrace{cc}_{=}\ldots$

   For example, 11 is doubly because it is formed by a pair of equal digits: 11. The number 1100 is also a doubly number because it is formed by 11 and followed by 00.

   The number 442222 is also doubly, but the number 444222 is NOT a doubly number because when digits are grouped, you get 44 42 22, and the digits of 42 are not equal. The number 110 is NOT doubly because it has an odd length. The number 9987 is NOT doubly because when you group 99 and 87, 87 has no equal digits.

   The first 10 doubly numbers are: $11, 22, 33, 44, 55, 66, 77, 88, 99, 1100$

   (a) (10 points) Write the function `isDoubly(n)`, which takes a positive integer `n` and returns `True` if `n` is a doubly number and `False` otherwise. You can write any additional helper functions that you need.

(b) (10 points) Write the function `nthDoubly(n)` which takes a non-negative integer `n` and returns the nth doubly number. `nthDoubly(0)` should return `11`, the first doubly number. You may assume that your implementation of `isDoubly(n)` functions properly, even if yours does not.

5. (15 points) **Free Response:**

   Write the function `isValidRGBStr(s)` that takes in a string `s` and returns `True` if `s` is a valid RBG string and `False` if it is not.

   A valid RGB string is of the form `rgb(x,y,z)` or `RGB(x,y,z)` and satisfies the following constraints:

   - `x`, `y`, `z` represent integers between 0 and 255.
   - There are no white spaces within the string.

   For example:

```
# valid RGB strings
assert(isValidRGBStr('rgb(0,0,0)') == True)
assert(isValidRGBStr('RGB(255,128,4)') == True)
assert(isValidRGBStr('RGB(255,255,255)') == True)

#invalid cases
# ups, spaces inside
assert(isValidRGBStr('RGB(255,255,   255)') == False)
# wrong capitalization: Rgb is not valid
assert(isValidRGBStr('Rgb(255,255,255)') == False)
# It must be of the form rgb(x,y,z) or RGB(x,y,z)
assert(isValidRGBStr('(255,128,4)') ==False)
 # red component equal to 15112 is not valid. It must be between 0 and 255
assert(isValidRGBStr("rgb(15112,2,255)") == False)
# nonsense RGB string, It must be of the form rgb(x,y,z) or RGB(x,y,z)
assert(isValidRGBStr('color blue') == False)
```

   Hint: The following built-in `string` methods may be useful: `isdigit()` that checks if the characters are numerical, and `split`.

Answer space for Question 5

6. (10 points) **Free Response:** destructiveListInList

Write the function `destructiveListInList(a, b, n)` which destructively modifies `a` (without modifying `b`) by adding all the values of `b` between elements `a[n-1]` and `a[n]` and returns the usual value returned by destructive functions. When `n == 0`, it adds all values of `b` onto the front of `a`. You may assume `0 <= n <= len(a)`. When `n == len(a)`, it adds all values of `b` at the back of `a` (the function behaves like `a.extend(b)`).

So this code works:

```
L = [3, 4, 1, 2]
destructiveListInList(L, [7, 6], 2)
assert(L == [3, 4, 7, 6, 1, 2])

L = [1, 3]
destructiveListInList(L, [4, 2], 2)
assert(L = [1, 3, 4, 2])

A = [4, 5]
B = [7, 8, 9]
destructiveListInList(A, B, 1)
assert(A == [4, 7, 8, 9, 5])
assert(B ==  [7, 8, 9])
```

You may not import or use any module other than copy. You may not use any method, function, or concept that we have not covered this semester. We may consider additional test cases not shown here.

Answer space for Question 6