15-112
Spring 2022 Exam 1
February 17, 2022

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 = 15112` `print(x//100 + x%10)`	

`L = [6,1,4]`
`reversed(L)`
`print(L)`	

`L = [6, 1, 4]`
`L.reverse()`
`print(L)`	

`L = [6,3,2]`
`ans = []`
`for i in range(len(L)):`
`  ans += L[i:]`
`print(ans)`	

`a = [3,2,1,0]`
`for i in range(2):`
`  a[a[i]] = a[i]`
`print(a)`	

`s = "easy"` `print(s + "y" + s[0] +  s[::-1])`	

`x = "study"`
`y = "student"`
`while x[0] == y[0]:`
`  y = y[1:]`
`  x = x[1:]`
`print(y)`	

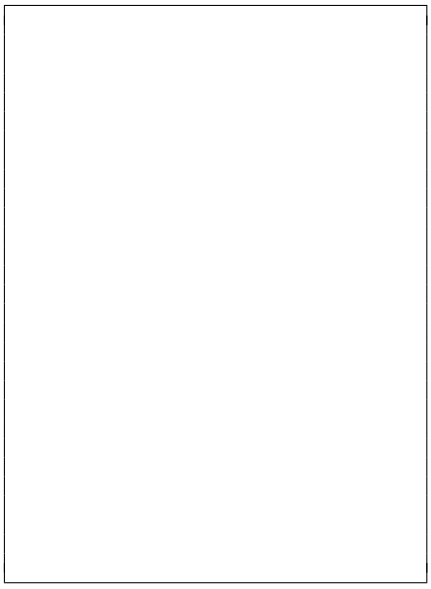
`s = "112rocks"`
`i = s.find("1",1)`
`print(s[i:])`	

`s = "4"`
`t = "2"`
`print(f"s+t{s + t}")`	

`def f(x):`
`  x = 3*x - 2`
`x = 2`
`f(x)`
`print(x)`	

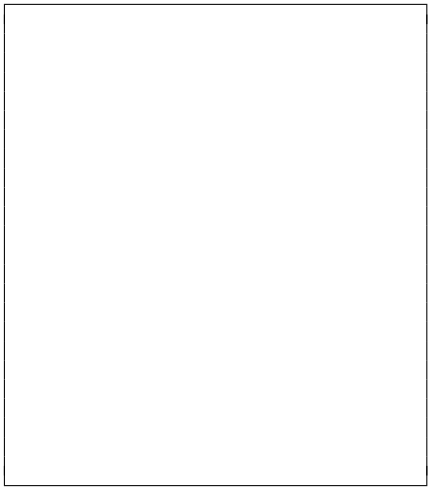
(b) (5 points) Indicate what the following program prints. Place your answers (and nothing else) in the box next to the code.

```
def ct1(a, b):
    for i in range(a, b, -1):
        j = -i
        i = abs(i)
        if i % 2 == 0:
            print("even", i)
        elif i > 0:
            print("foo", i)
        if j % 2 == 1:
            print("odd", j)
        if j > 0:
            print("almost done!",j)
    print("done!")
b = 2
a = -2
ct1(b,a)
```



(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):
    t = s
    s += s[:2]
    print("one", s)
    t += s[::-1]
    print("two",t)
    print(t.replace('cb','f'))
    print(t.count('b'))
print(ct2('abc'))
```



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

```
def f(a, b):
    b += ["hi"]
    a = b[:2]
    return a+b

def ct3(a):
    b = a
    c = b[:]
    b[0] = 42
    a.append("wow")
    b = f(a, b)
    if 42 in c:
        c.remove(42)
    print("a1",a)
    a[-2] = 112
    print("a2:", a)
    print("b2:", b)
    print("c2:", c)
z = ["feb", 17]
ct3(z)
print("z:", z)
```

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) == 2 or x[1] == 112`		N/A
`x%y == 1 and x//y == 1`		
`len(x) == 1 and x.upper() == x.lower()`		N/A
`len(x) == 3 and x.split() == ['4','2']`		N/A
`(y // x == x // y - 1)`		
`"racecar"[x:] == "racecar"[x::-1]`		N/A
`len(y) == 2 and x == y.pop(1)`		
`len(x) == 2 and x in "15112"[3:]`		N/A
`int(x) // 10 == int(x) % 10`		N/A
`len(x) == 2 and ord(x[1]) == ord(y[0])`		

(b) (5 points) Find the arguments 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.

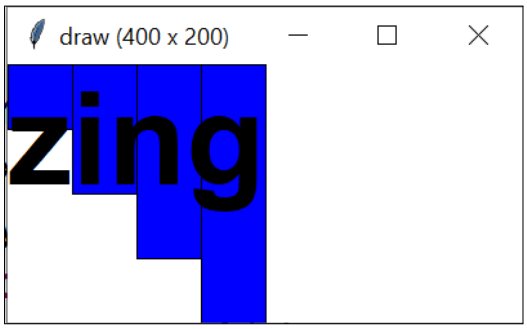
```
def rc1(n):
    assert(n < 1000)
    a = 0
    while n != 0:
        d = n%10
        a = a*10 + d
        if d%2 == 0:
            return 100 < a and a < 1000
        n //= 10
    return False
```

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

```
def rc2(L):
    if not isinstance(L, list) or len(L) != 3:
        return False
    t = ""
    while(L != []):
        s = L.pop()
        if type(s) != str:
            return False
        t += s[::-1]
    return t == str(15112)
```

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 = 50 # Width of first rectangle
    dy = 50 # Height of first rectangle
    for i in range(4):

        x0   = _________________ # (A)

        y0   = 0

        x1   = _________________# (B)

        y1   = _________________# (C)

        col  = _________________# (D)

        canvas.create_rectangle(x0, y0, x1, y1, fill=col)

    canvas.create_text(200, 0, text='Amazing',

                       anchor=_________________# (E)

                       font='Arial 50 bold')

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

4. **Free Response:** Trine 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 *trine* number (coined term) if it is a positive integer such that...

   - Somewhere in the number some digit occurs consecutively 3 times or more (so the same digit occurs 3 times or more in a row, ignoring leading zeros),
   - The number contains at least 3 digits.

   For example, 222 is a trine number because the digit 2 occurs three times in a row. The number 2000 is also a trine number because 0 occurs three times in a row. The number 44440 is also a trine number, but the number 44044 is NOT a trine number because no digit occurs three or more times in a row.

   The first 5 trine numbers are: 111, 222, 333, 444, 555.

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

(b) (10 points) Write the function `nthTrine(n)` which takes a non-negative integer `n` and returns the nth trine number. `nthTrine(0)` should return `111`, the first trine number. You may assume that your implementation of `isTrine(n)` functions properly, even if yours does not.

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

   Write the function `extractIntegerNumbers(s)` that takes a string `s` and returns a list that contains all the integers that occur in `s`. You may assume that the only numbers in the text are integers and that numbers are always composed of consecutive digits (without commas, for example).

   Note: To receive any credit, you must not use sets, dictionaries, or other concepts we have not covered in the notes this semester.

   Here are some test cases:

```
assert(extractIntegerNumbers("3 dogs, 17 cats, and 14 cows!") ==[3,17,14])
assert(extractIntegerNumbers("I saw four dogs")==[])
assert(extractIntegerNumbers("cmu15112 is the best!")==[15112])
assert(extractIntegerNumbers("Five -3 equals 2")==[-3,2])
assert(extractIntegerNumbers("be careful with neg. ints 32-42")==[32,-42])
assert(extractIntegerNumbers("cmu15-112isGr8")==[15,-112,8])
```

Answer space for Question 5

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

Write the function `prepend(a, b)` which destructively modifies `a` (without modifying `b`) by adding all the values of `b` onto the front of `a`, and returns the usual value returned by destructive functions. So this code works:

```
a = [1, 2]
b = [3, 4]
prepend(a, b)
assert((a == [3,4,1,2]) and (b == [3,4]))
```

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 use additional test cases not shown here.

Answer space for Question 6