15-112
Fall 2024 Exam 1
October 1, 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 and string are imported; do not import any other modules.
- **Do not use these post-midterm 1 topics/constructs: lists, sets, maps/dictionaries, recursion, or classes/OOP.**

Don't write anything in the table below.

Question	Points	Score
1	30	
2	20	
3	15	
4	15	
5	20	
6	0	
7	0	
Total:	100	

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

   (a) (7 points) CT0

```python
def ct0():
    m = 72
    n = m / 1
    print(type(m) == int, type(n) == int)
    x = int(2 * str(2 * int("5"))) + 2
    b = x % 2 == 1
    print(x, b)
    print(int(b), int(not b))
    print(1 + 8 // 3 * 2 - 3)
    c = math.ceil(8/3)
    d = math.floor(8/3)
    print(c, d)

ct0()
```

   (b) (8 points) CT1

```python
def ct1(a, b):
    c = 1
    for i in range(1, a // b):
        if i % (a % b) == 0:
            print(i, c)
            c = c * i
    a = a / 2
    b = b / 3
    return c

a = 3
c = 20
print(ct1(c, a))
print(a, c)
```

(c) (7 points) CT2

```
def ct2(s, x):
    for i in range(len(s)):
        print(i, s, x)
        if s[i] == x[-1]:
            s += x[-1]
            x = x[: len(x) - 1]
        x = x[::-1]
    return s

print(ct2("ponder", "roednp"))
```

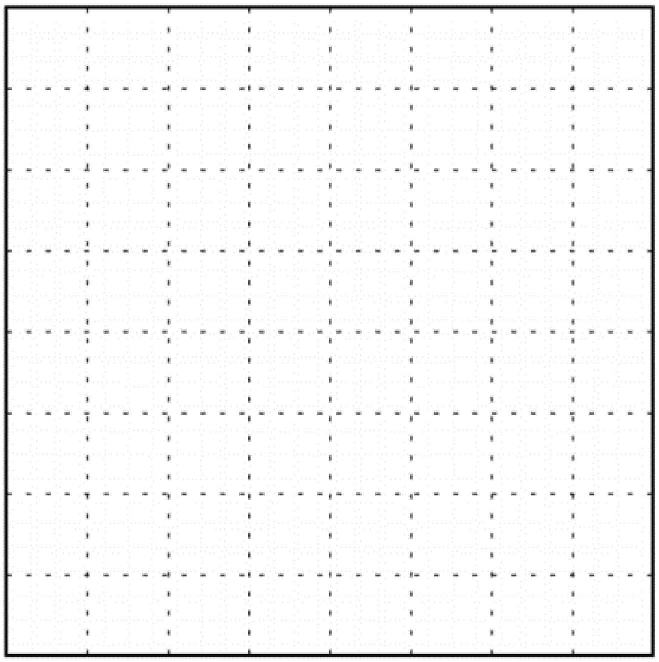
(d) (8 points) CT3

Do not use a pen while solving this question. Use a pencil.

```
def drawCT(app):
    a = app.width//4
    b = app.height//8
    drawRect(a, b, 3 * a, 3 * b, fill=None, border="black")
    drawCircle(a, 4*b, a//2, align="top", fill=None, dashes=True, border="black")
    drawLabel("ABC", 2*a, 3*b, rotateAngle=-90, align="bottom")

    cx = app.width//2 + 100
    cy = app.height//2 + 50
    w = 100
    h = 200
    color = "black"
    for i in range(0, 2):
        drawOval(cx, cy, w, h, fill=color, border="black")
        w, h = h, w
        color = "white"

def redrawAll(app):
    drawCT(app)

runApp(width=400, height=400)
```

Given that the box below is your canvas, with a width and height of 400 each, draw what the above code would display.

Hint: Each of the small boxes on the canvas is 50x50 pixels.



2. **Free Response**: Mountain Numbers

   Note: For this problem, you **may not use strings**, lists, sets, maps/dictionaries, recursion, or classes/OOP.

   A *mountain number* (coined term) is a 3-digit or more number with an odd number of digits where the digits strictly ascend in value to the middle digit of the number, and then strictly descend in value to the end.

   For example, `1235420` is a mountain number. The middle digit is 5, and the digits leading up to 5 are 1, 2, 3 which continually ascend to 5. The digits following 5 are 4, 2, 0 which continually descend from 5.

   Here are some other mountain numbers: 154, 13764, and 4789720.

   The following are *not* mountain numbers:

   - 5 (not at least three digits)
   - 1235220 (5220 is not a strictly descending sequence)
   - 1255420 (1255 is not a strictly ascending sequence)
   - 123421 (not an odd number of digits)

   (a) (14 points) Write the function `isMountain(num)` which, given a number `num` returns `True` if `num` is a mountain number and `False` otherwise.

(b) (6 points) Write a function `nthMountain(n)` which takes as an input a non-negative integer `n` and returns the nth mountain number. Consider the following testcases:

```
assert nthMountain(0) == 120
assert nthMountain(1) == 121
assert nthMountain(2) == 130
assert nthMountain(3) == 131
assert nthMountain(4) == 132
assert nthMountain(5) == 140
assert nthMountain(100) == 365
assert nthMountain(9989) == 6789876
```

Note: For this problem, you may assume that your answer to part (a) works, even if yours does not.

3. (15 points) **Free Response**: Column Order

Write the function `columnOrder(s)` which, given a string `s` whose length is a perfect square (meaning $len(s) == n * n$ for some integer n), returns the characters of that string rearranged into *column order* (coined term). Column order is the order they would be read out if the characters of `s` were stored in an nxn grid and the columns traversed from left to right.

For example, consider the case where `s = "abcdefghijklmnop"`:

1. `s` has a length of 16, so $n = 4$.
2. Writing `s` in a $4x4$ grid would yield:

```
a b c d
e f g h
i j k l
m n o p
```

3. Column order involves reading each column one at a time. The far left column is `"aeim"`, the next column is `"bfjn"`, etc.
4. The column order return from the function would be `"aeimbfjncgkodhlp"`

Consider the following test cases:

```
assert columnOrder("abcd") == "acbd"
assert columnOrder("abcdefghi") == "adgbehcfi"
assert columnOrder("abcdefghijklmnop") == "aeimbfjncgkodhlp"
assert columnOrder("IV1LE2O1!") == "ILOVE112!"
```

Note that you may assume that `len(s)` is a perfect square, you don't need to verify.

4. (15 points) **Free Response**: Reversing Numbers

   Write the function `reverseNumbers(s)` which, given a string `s` returns that same string, but with all numbers in the string having their digits reversed. A number is defined as a consecutive set of digits without non-digit characters between them.

   Consider the following test cases:

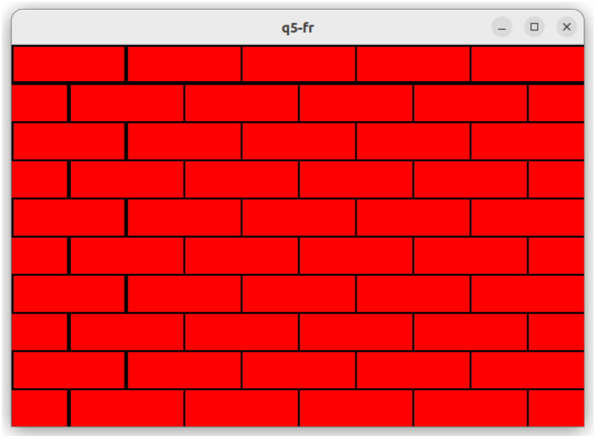
```
assert reverseNumbers("1234") == "4321"
assert reverseNumbers("Blue 1234 Red319") == "Blue 4321 Red913"
assert reverseNumbers("Blue 1.234 Red31.9") == "Blue 1.432 Red13.9"
assert reverseNumbers("0786ThereAre123CheeseSlicesOn7SandwichesInThe12Stores.4123") == \
                      "6870ThereAre321CheeseSlicesOn7SandwichesInThe21Stores.3214"
```

5. **Free Response**: Pretty Wall

In this problem, you will write two helper functions and then use them together to draw a picture.

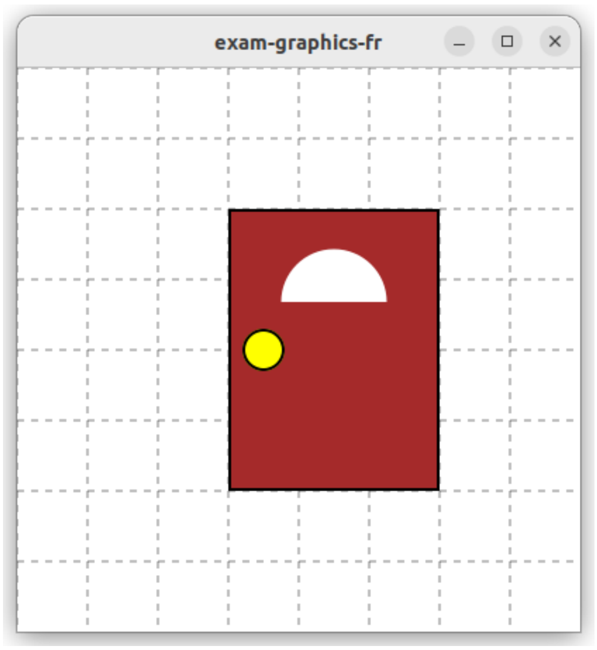
(a) (8 points) Write the function `drawBricks(app, r, b)`, which draws a brick wall filling the canvas. There should be `r` rows of bricks and each row should have `b` bricks. Every-other row should be offset by 50%.

For example, a call to `drawBricks(app, 10, 5)` on a 600x400 canvas produces the following output:



(b) (7 points) Write the function `drawDoor(x, y, width, height)` which draws the door as pictured below.

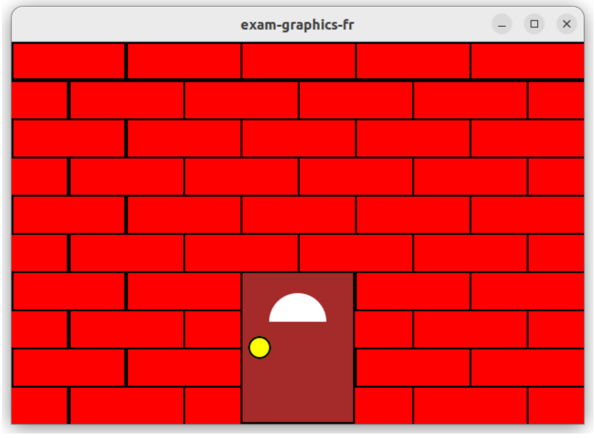
For example, calling `drawDoor(150, 100, 150, 200)` on a 400x400 canvas produces the following output:



Note: The size of the window and the door knob should scale in proportion to the size of the door. When choosing their exact size and location on the door, do your best, but don't obsess.

(c) (5 points) Write the function `drawPrettyWall(app)` which draws a brick wall with a door, as seen below.

For example, calling `drawPrettyWall(app)` with a 600x400 canvas draws the following:



Notes:

- The location and size of the door does matter.
- The image should look correct for a variety of canvas sizes. (So don't assume it will always be 600x400.)
- You may assume that you have working versions of `drawBricks(app, r, b)` and `drawDoor(x, y, width, height)`, even if yours do not work.

6. (1 point (bonus)) **Bonus:** Deep Thoughts

   Go back to your answer to Question 2 part b and modify it to account for the fact that there is not an infinite quantity of mountain numbers.

7. (2 points (bonus)) **Code Tracing**

```
def f(c1, c2):
    return abs(ord(c1) - ord(c2))

def g(a, b):
    if b - a < 0:
        return -1
    else:
        return 1

def ct4(x, z):
    r = 0
    a = 0
    b = 0
    # Hint: You need to figure out what this does and not just code trace
    for c in x:
        if f(x[0], c) >= 0 and f(x[0], c) < 10:
            p = z[a : b : g(a, b)]
            if p != "":
                print("Go:", p)
                r += int(p)
            a = b
            b = f(x[0], c)
    return r

print(ct4("KOVMTAZEB", "123456789"))
```