15-112
Spring 2024 Exam 1
February 15, 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, string, and copy 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	10	
2	10	
3	10	
4	10	
5	20	
6	15	
7	25	
Total:	100	

1. (10 points) **Code Tracing**: Powerful Mods

   Place your answer (and nothing else) in the box below the code.

```
def f(a, x):
    return 10 * a + x

def g(a):
    return a % 3 == 0

def ct1(n):
    r = 0
    for i in range(n, n**2, 2):
        if g(i):
            continue
        elif i // 10 > 0:
            break
        else:
            r = f(r, i)
    return r

print(ct1(3))
```

2. (10 points) **Code Tracing**: Stringin' Around

   Place your answer (and nothing else) in the box below the code.

```
def ct2(s1, s2):
    s2 = s2.upper()
    r = ""
    for i in range(len(s1)):
        if s1[i].isdigit() and s1[i+1].isdigit():
            r += s2[i:int(s1[i]):int(s1[i+1])]
        else:
            r += s1[i]
    return r

print(ct2("31a82b","welcometothejungle"))
```

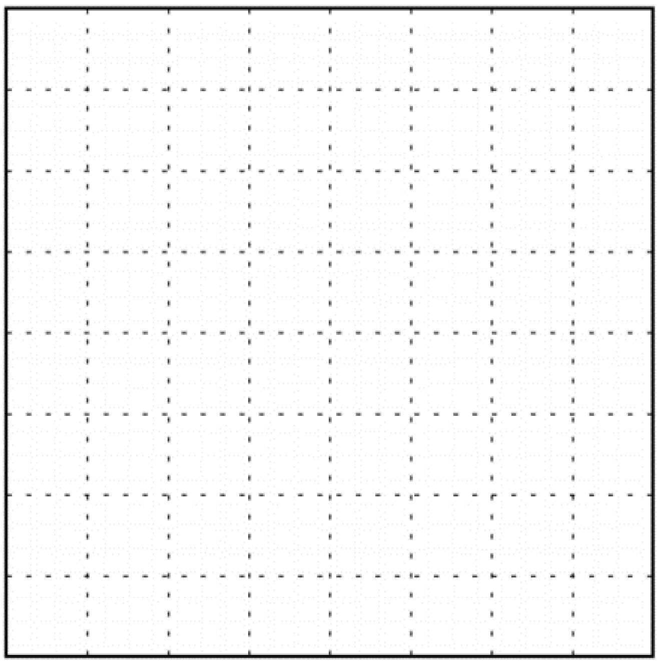
3. (10 points) **Code Tracing**: Graphics!

   Note: Use a pencil when solving this problem. Trust us.

   Given that the box below is your canvas, with a width and height of 400 each, draw what the following code will display. You can assume that this is called within the appropriate graphics helper code.

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

```
def redrawAll(app):
    step = app.width//4
    for i in range(2):
        for j in range(2):
            color = "white" if (i + j) % 2 == 1 else "black"
            drawCircle(step + 2 * step * i, step + 2 * step * j, step / 2,
                       border="black", fill=color)
    drawRect(step, step, 2 * step, 2 * step, border="black", fill="white")
    drawLine(0, app.height, app.width, 0)
    drawLine(step, step, 2 * step, 2 * step)
```



4. (10 points) **Code Tracing**: What the String?!

   Place your answer (and nothing else) in the box below the code.

```
def ct4(s1):
    s2 = s1
    s3 = ""
    for c in s1:
        if c.isspace():
            s2.replace(c, "X")
        elif c.isupper():
            s3 += c.lower() + str(s1.upper().count(c))
    print(s2)
    return s3

print(ct4("0a A2dR5"))
```

5. **Free Response:** Sperjish Numbers

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

   A sperjish number (coined term) is a number with a least three digits, all digits are either 0 or odd, and the sum of the digits is a multiple of the number of digits.

   As an example of a sperjish number, consider 109350:

   - 109350 is at least 3 digits.
   - Each of the digits in 109350 is either 0 or odd.
   - The sum of the digits ($1 + 0 + 9 + 3 + 5 + 0 = 18$) is a multiple of the number of digits (6).

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

(b) (6 points) Write a function `nthSperjish(n)` which takes as an input a number `n` and returns the nth sperjish number. Consider the following testcases:

```
assert nthSperjish(-1) == None
assert nthSperjish(0) == None
assert nthSperjish(1) == 105
assert nthSperjish(2) == 111
assert nthSperjish(3) == 117
assert nthSperjish(100) == 1591
assert nthSperjish(1000) == 51711
```

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

6. **Free Response:** Word Builder

   We say that a word, w1, can be used to build another word, w2, if some subset of the letters from w1 can be rearranged to produce the word w2. For example, the word "water" can be used to build "rate", "tear", "at", etc. (But it can't be used to build "rater" because "water" only has one r, not two.)

   (a) (8 points) Write the function `wordCanBeBuilt(s1, s2)` which returns `True` if the string `s1` can be used to build the string `s2`, and `False` otherwise.

(b) (7 points) Write the function `canBuild(word, manyWords)` that takes as arguments a string containing a word and another string containing a comma-separated list of words, and returns a comma-separated list of words from `manyWords` that can be built from `word`.

Consider the following test cases:

```
assert canBuild("damthewater", "ad,add,sad,mad,tear") == "ad,mad,tear"
assert canBuild("freakymonkeytips", "far,jet,pit,lip,jaw,sip") == "far,pit,sip"
bigWord = "supercalifragilisticexpialidocious"
bigList = "jet,pit,hero,sail,space,lack,pride,tile,park,drug,steam"
assert canBuild(bigWord, bigList) == "pit,sail,space,pride,tile,drug"
```

Notes:

- Remember that you can't use lists on this exam.
- `.split()` may be very useful here but you may only loop over the result, and may not index/slice the result or use list functions or list methods.
- You may assume that you have a version of `wordCanBeBuilt(s1, s2)` that works, even if yours from part (a) does not work.

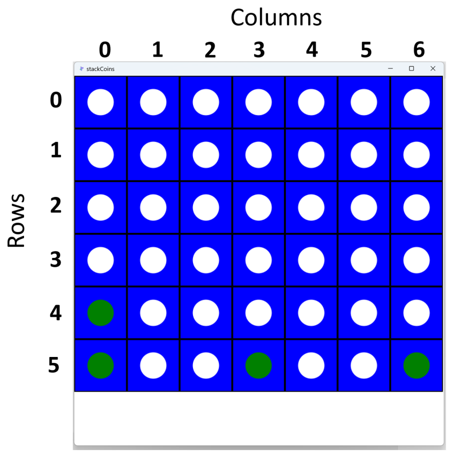
7. **StackCoins Board Game**

(a) (10 points) **Fill-in The Blanks:** Graphics

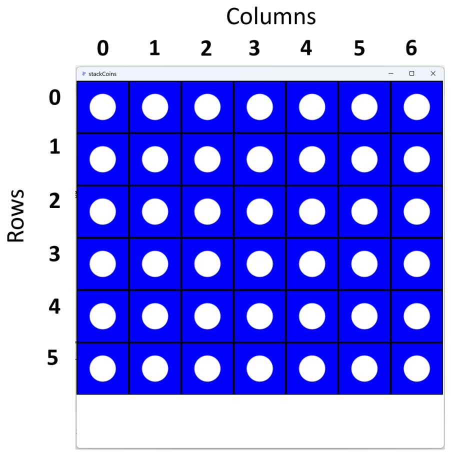
Fill in the code for the function `drawStackCoins(app,coinCells)`, which draws a 6x7 board game, with a total of 42 cells. The function takes as an input the string `coinCells` that specify the cells at which green coins are currently inserted. Each cell is represented as two numbers that map to rowID 0,1,2,...,5 and columnID 0,1,2,...,6; where cell(00) is the corner left top cell in the board.

For example, `drawStackCoins(app, "02,25")` draws two green coins/circles; one at cell 02 (i.e. row0 column2) and the other at cell 25 (i.e. row2 column5).

Consider the following examples (note that the printed row and column numbers are there to help your understanding, they are not present in the actual game):



(a) `drawStackCoins(app, "50,40,53,56")`



(b) `drawStackCoins(app, "")`

You can assume that `app.width==app.height`. The cells should be drawn to span the canvas width. Each cell should be drawn as a blue square with a black border and a circle centered in the square. The color of the circle depends on whether the cell is empty (`'white'`) or has a (`'green'`) coin. Your code should be able to handle window resize. You can assume that `cmu_graphics` is already imported and there is a function `redrawAll(app)` that will call your function.

```
def drawStackCoins(app, coinCells):
    numRows = 6
    numCols = 7
    sideL = app.width // numCols

    for r in range(numRows):
        for c in range(numCols):


            x = ___________________________________________________________________


            y = ___________________________________________________________________


            drawRect(x, y,_________________________________________________________)


            if _________________________________ in ________________________________:
                circleColor = 'green'
            else:
                circleColor = 'white'


            drawCircle(____________________________________________________________)
```

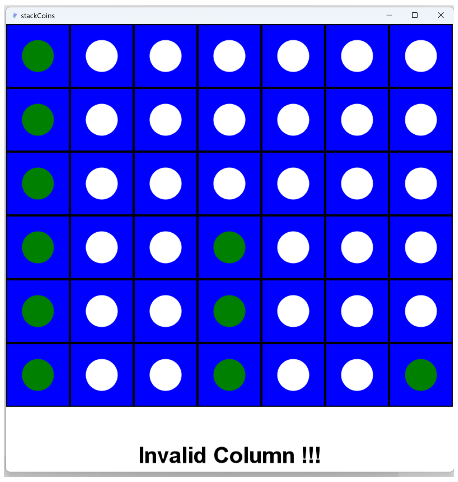
(b) (15 points) **Free Response:** Animations

Now, Let's animate the StackCoins board game.

In this part assume the function `drawStackCoins(app, coinCells)` already exists and works well. Do not write it again, instead just call it when you need it.

The game should have the following features:

1. The game starts with an empty board (all cells have white circles); See figure (b) from the previous problem.
2. A coin is inserted in a column by pressing at any place on that column.
3. A coin should be stacked on the selected column. Meaning it should be inserted on the next empty cell in that column. For example: the next empty cell in column 3 in Figure 2 is cell (`"23"`) since cell (`"33"`) is filled. *Remember to update the string coinCells to keep track of inserted coins.*
4. If the user presses on a column that is already full, a message should be displayed on the empty space at the bottom of the canvas saying: *Invalid Column !!!*; See figure 2 below.
5. The game can be reset by pressing the r key.



Invalid Column !!!

Figure 2: The view after trying to insert a coin in a full column (column 0)

Notes:

- **Design your helper functions wisely and the problem will be easier.**
- `.split()` may be useful here but you may only loop over the result, and may not index/slice the result or use list functions or list methods.
- Do not hardcode for a 400x400 canvas. However, you may assume the canvas is square and at least 100x100 pixels.
- You will be penalized if your code results in an MVC violation.
- Make reasonable choices for anything not specified above.
- To solve this, you need to write onAppStart, onKeyPress, onMousePress, and redrawAll.
- You do not need to include any imports or the main function.

Do not write your answer on this page. Write your answer on the next two pages.

Additional space for Question 7(b).

Additional space for Question 7(b).