15-112 Spring 2021 Exam 1

Name:

Andrew ID:

- You may not use any books, notes, or electronic devices during this exam.
- You may not ask questions about the exam.
- Show your work on the exam to receive credit.
- You may use the backs of pages as scratch paper. Nothing written on the back of any pages will be graded.
- 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 dictionaries, sets, try/except, or recursion on this exam. There may be other restrictions on individual problems.

1. Summarize the Code In **two sentences or less**, describe, at a high-level, what each piece of the following functions do. The first one is done for you as an example so that you understand what sort of things to write.

Note: The "two sentences or less" restriction will be strictly enforced. (This is for your own good. . .)

Sample

```
def sampleFunc(p,q):
    return math.isclose(p/q,p//q)
```

Answer: Determines whether or not `p` divided by `q` is a whole number.

Part A [5 pts]

```
def summarizePartA(n):
    if n < 15:
        return None
    else:
        return n%(n-7)
```

Part B [5 pts]

```
def summarizePartB(f):
    mypic = ImageWriter.loadPicture(f)

    width = ImageWriter.getWidth(mypic)
    height = ImageWriter.getHeight(mypic)

    a = []
    for i in range(width):
        for j in range(height):
            colors = ImageWriter.getColor(mypic, i, j)
            a.append(colors[0])
    b = sum(a)/len(a)

    for i in range(width):
        for j in range(height):
            colors = ImageWriter.getColor(mypic, i, j)
            colors[0] = b
            ImageWriter.setColor(mypic, i, j, colors)

    ImageWriter.savePicture(mypic, f)
```

2. Free Response – Number Play [15 pts] **Note:** You may not use strings or lists in this problem. If you do, your score on this problem will receive a 10 point deduction.

We will say that a positive integer is tennish (a coined term) if its digits sum to 10. So, 127, 721, and 100200007 are all tennish.

With this in mind, write the function `nthTennish(n)` that takes a non-negative int n and returns the nth tennish number. Note that `nthTennish(1)` returns 19, which is the smallest tennish number.

3. Free Response – Paths [15 pts] We will say that a string is a path if it only contains the letters U, D, L, or R, which indicate to take one step in the directions up, down, left, and right, respectively. Two paths are equivalent if, when starting at (0,0) and following those paths, you end up at the same point.

For example, the path `"URUL"` goes from (0,0) up to (0,1), right to (1,1), up to (1,2), and left to (0,2). The path `"UU"` goes from (0,0) up to (0,1) and up to (0,2). So `"URUL"` and `"UU"` are equivalent paths, as both end at (0,2). Also, note that it is fine for paths to move through negative indexes.

With this in mind, write the function `areEquivPaths(p1, p2)` that returns `True` if `p1` and `p2` are strings representing equivalent paths, and `False` otherwise. From the example above, `areEquivPaths("URUL", "UU")` would return `True`.

You may assume that `p1` and `p2` will be non-empty strings that contain only the characters U, D, L, or R.

Hint: You should probably write a helper function that calculates and returns the final location of a path.

4. Free Response – Get Important Words [15 pts] We will consider a word in some text to be an important word if it starts with a capital letter. The end of an important word is signified by either a space, a comma, a period, or the end of the text.

For example, consider the string, `"The Quick Red, foxJumped over the lazy brown Dog.and got Away"`. The important words in this strings are: "The", "Quick", "Red", "Jumped", "Dog", and "Away".

With this in mind, write the function `getImpWords(s)` that returns a list of all of the important words in `s`. From the example above, `getImpWords("The Quick Red, foxJumped over the lazy brown Dog.and got Away")` should return `['The', 'Quick', 'Red', 'Jumped', 'Dog', 'Away']`

5. Free Response – Nearly Sorted [20 pts] For the following problem, assume the existence of the following helper function:

```python
# Swap elements i and j in list L
def swap(L, i, j):
    tmp = L[i]
    L[i] = L[j]
    L[j] = tmp
```

Now, on to the problem: We will say that a list is "nearly-sorted" (a coined term) if it is not sorted but it requires exactly one swap of two of its values to become sorted from least to greatest.

For example...

- `a=[3,13,7,10,4]` is nearly-sorted, since calling `swap(a,1,4)` results in a sorted list.
- `a=[1,2,30]` is not nearly-sorted, since it is already sorted.
- `a=[9,7,4,1,2,3]` is not nearly-sorted, since there is no single swap of two elements that can result in it being sorted.

With this in mind, write the function `checkNearlySorted(L)` that takes a list `L` of integers, and returns `False` if the list is not nearly sorted. If the list is nearly sorted, the function does not return `True`, but rather it returns the tuple `(i,j)`, where i<j and calling `swap(L,i,j)` would make the list sorted. So, from the example above, `checkNearlySorted([3,13,7,10,4])` should return `(1,4)`. For full credit, your function must be non-destructive. Also, do not worry about how efficient your algorithm is.

Extra space for the nearly sorted question.

6. Code Tracing (Part 1) [6 pts] Indicate what the following program prints. **Put a box around your final answer.**

Hint: There are six lines of output.

```
def ct1(L):
    ret = []
    a = 0
    for item in L:
        if isinstance(item, str):
            for c in item:
                if c.isdigit():
                    ret.append(int(c)+5)
                    print("L1", ret)
            ret.insert(0,c)
            print("L2", ret)
    for item in ret:
        if isinstance(item, int):
            a += item
    print(a)
    return a

print(ct1([5,"78","QA",32]))
```

7. Code Tracing (Part 2) [9 pts] Indicate what the following program prints. **Put a box around your final answer.**

```
def fa(d,e,f):
    d = ["A","B","C"]
    e[3] = 43
    f[3] = 44
    print("d",d)
    print("e",d)
    print("f",d)

def ctRef():
    a = [10,11,12,13]
    b = a[:]
    c = b
    a[0] = "Bob"
    b[1] = 50
    c[2] = 60
    print("a1",a)
    print("b1",b)
    print("c1",c)
    fa(a,b,c)
    print("a2",a)
    print("b2",b)
    print("c2",c)

ctRef()
```

8. Reasoning Over Code [10 pts] For the following function, find a value `x` which will cause `roc1(x)` to return `True`. **Put a box around your final answer.**

```
def roc1(x):
    assert isinstance(x,str)
    a = 0
    r = ""
    for item in x:
        if item.isupper():
            a += 3
            r += item.lower()
        elif item.islower():
            a += 4
        elif item in "Hello,Jim":
            a += 1000
    return r[::-1] == "kitty" and a == 3031
```