Eric Roberts
CSCS 121

Handout #23
October 1, 2018

Practice Midterm Examination #2

Problem 1: Simple Python expressions (10 points)

Compute the value of each of the following expressions and write it in the space provided. If the line produces an error, write "Error" in this space. Assume that the constant `ALPHABET` has been initialized as in the reader and has the following internal value, listed with both positive and negative indexes:

`ALPHABET`

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
−26	−25	−24	−23	−22	−21	−20	−19	−18	−17	−16	−15	−14	−13	−12	−11	−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

1a) `1 * 2 % 3 + (4 % 5 * 6) * 7 * 8 // 9 * 10` _____

1b) `str(5 * 2) + str(5 + 1) * 2` _____

1c) `ALPHABET[-9::-13] + ALPHABET[4:2:-1]` _____

1d) `ALPHABET[len(ALPHABET)]` _____

Problem 2: Program tracing (10 points)

What output does the following program produce:

```
# File: Mystery.py

def mystery(x):
    def enigma(z):
        nonlocal y
        y -= 2
        return z[::6] + z[y]

    y = len(x)
    z = x[1 - y]
    print(z)
    z += enigma(x)
    print(z)
    z += enigma(x)
    print(z)

# Startup code

if __name__ == "__main__":
    mystery("abcdefgh")
```

Problem 3: Simple Python programs (15 points)

A ***spoonerism*** is a phrase in which the leading consonant strings of the first and last words are inadvertently swapped, generally with comic effect. Some examples of spoonerisms include the following phrases and their spoonerized counterparts (the consonant strings that get swapped are underlined):

crushing blow → blushing crow
sons of toil → tons of soil
pack of lies → lack of pies
jelly beans → belly jeans
flutter by → butter fly

In this problem, your job is to write a function

```
def spoonerize(phrase):
```

that takes a multiword phrase as its argument and returns its spoonerized equivalent. For example, you should be able to use your function to duplicate the following console session in which all the examples come from Shel Silverstein's spoonerism-filled children's book *Runny Babbit:*

IDLE

```
>>> from Spoonerize import spoonerize
>>> spoonerize("bunny rabbit")
runny babbit
>>> spoonerize("silly book")
billy sook
>>> spoonerize("take a shower")
shake a tower
>>> spoonerize("wash the dishes")
dash the wishes
>>>
```

In this problem, you are not responsible for any error-checking. You may assume that the phrase passed to **`spoonerize`** contains nothing but lowercase letters along with spaces to separate the words. You may also assume that the phrase contains at least two words, that there are no extra spaces, and that each word contains at least one vowel. What your method needs to do is extract the initial consonant substrings from the first and last words and then exchange those strings, leaving the rest of the phrase alone.

Hint: Remember that you can use methods from the book. The **`findFirstVowel`** and **`isEnglishVowel`** methods from the Pig Latin program will certainly come in handy.

Problem 4: Using the Portable Graphics Library (20 points)

When Marissa Mayer (now CEO of Yahoo! Inc.) took Stanford's version of CSCI 121, her entry in the Graphics Contest was a screensaver program that simulated a fireworks show. Your task in this problem is to implement an exam-sized subset of her contest-sized application.

Your program should execute the following steps:

1. Create a tiny dot (an unfilled circle whose width and height are both one pixel) at the point that lies at the center of the bottom of the window and color it using a randomly chosen color.
2. Choose a random point somewhere in the top half of the window.
3. Animate the motion of the dot so that it moves to the point you chose in step 2. The dot should move so that gets to its destination in `FLIGHT_TIME` milliseconds.
4. Once the dot reaches its destination, you should change the behavior of the animation so that the circle radius increases by `DELTA_RADIUS` pixels for `EXPANSION_TIME` milliseconds.

An animation with random behavior is difficult to represent on the printed page, but the final image on the window will look something like this (the dotted line shows the flight path of the dot, although this line would not appear on the window):