Eric Roberts
CS 151

Handout #14
October 6, 2020

Practice Midterm Examination #1

This handout is intended to give you practice solving problems that are comparable in format and difficulty to those which will appear on the midterm examination next Tuesday, October 13. A solution set to this practice examination will be posted on Thursday, when I will hand out a second practice exam.

Time and place of the exam

The midterm will run from 8:00 to 9:30 A.M. in Ford 122 (our regular classroom). If any of you need to take the exam remotely or if you need special accommodations, please let me know by Friday at noon so that I can make the necessary arrangements.

Coverage

The exam covers the material presented in class through the lecture on September 29, which means that you are responsible for the material in Chapters 1 through 6 of the reader.

General instructions

Answer each of the four questions included in the exam. Write all of your answers directly on the examination paper, including any work that you wish to be considered for partial credit. (Note: The actual exam will have more room for your answers and for any scratch work.)

Each question is marked with the number of points assigned to that problem. The total number of points is 55. I intend for the number of points to be roughly comparable to the number of minutes you should spend on that problem. I've set the time limit for the exam to 90 minutes to ensure that you have enough time to check your work and recover from false starts.

In all questions, you may include functions or definitions that have been developed in the course simply by using supplying their name. You don't need to write `import` statements, since we will know where those functions come from.

Unless otherwise indicated as part of the instructions for a specific problem, comments are not required on the exam. Uncommented code that gets the job done will be sufficient for full credit on the problem. On the other hand, comments may help you to get partial credit if they help us determine what you were trying to do.

The examination is open-book, and you may make use of your own notes, the course handouts, and the reader. You may not, however, use a computer of any kind.

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) `9 * 8 * (7 // 6 + 5) * 4 + 3 - 2 * 1`

1b) `7 < 9 - 5 and 3 % 0 == 3`

1c) `ALPHABET[5:15:3]`

1d) `ord("F") - ord("A")`

Problem 2: Program tracing (10 points)

What output does the following program produce:

```
# File: Mystery.py

def mystery(x):
    def enigma(s, t):
        t -= 2
        return s[::6] + s[t]

    y = len(x)
    z = x[1 - y]
    print(z)
    z += enigma(x, y)
    print(z)
    z += enigma(x, y - 2)
    print(z)

# Startup code

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

Problem 3: Python programs (15 points)

Write a function

```
def longest_character_run(s)
```

that takes a string `s` and returns the longest substring that consists of a single repeated character. For example, calling

```
longest_character_run("1066")
```

should return `"66"` because there are no longer runs of matching characters. Similarly, calling

```
longest_character_run("AABCCCCAAABB")
```

should return `"CCCC"`. There are more instances of the character `"A"`, but they don't appear in a single consecutive run.

If more than one subsequence has the same maximal length, your function should return the one that appears earliest in the string. Thus, calling

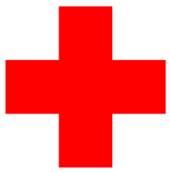
```
longest_character_run("XXXXYYZZZZ")
```

should return `"XXXX"` and not the equally long substring `"ZZZZ"`.

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

Write a graphical program that does the following:

1. Creates the following cross as a `GCompound` containing two filled rectangles:



   The color of the cross should be red, as in the emblem of the International Red Cross. The horizontal rectangle should be 60×20 pixels in size and the vertical one should be 20×60. They cross at their centers.
2. Adds the cross to the graphics window so that it appears at the center.
3. Moves the cross at a speed of 2 pixels every 20 milliseconds in a random direction, which is specified as a random real number between 0 and 360 degrees.
4. Every time you click the mouse inside the cross, its direction changes to some new random direction—again chosen as a random real number between 0 and 360 degrees—but its velocity remains the same. Clicks outside the cross have no effect.

If you were actually to write such a program, you would presumably supply some means of making it stop, such as when the cross moves off the screen. For this problem, just have the program run continuously without worrying about objects moving off screen or how to stop the timer.