15-110 Hw2 - Written Portion

Name:

AndrewID:

#1 - Gates and Circuits Translation - 5pts

Given the truth table shown below, construct a circuit that produces the same result as the Result column in the table. You may either use logic.ly, or you may draw the circuit by hand using the shorthand shown in the slides. **Hint:** try to find a pattern in the Result that you can map to patterns created by the basic gates we discussed.

X	Y	Z	Result
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

Click Here to Add Image

#2 - Gates and Circuits Construction - 10pts

Identify a boolean equation that takes three inputs, X, Y, and Z, and returns True if exactly two of the three values are True. You must represent this equation as a boolean expression, a Truth Table, and a circuit. For the circuit, you may either use logic.ly (or another online circuit simulator), or draw the circuit by hand using the shorthand shown in the slides.

Boolean Expression: _____

Truth Table

Click Here to Add Image

Circuit

Click Here to Add Image

#3 - Code Tracing with For Loops - 5pts

In twenty words or less, describe what the function below does. You may assume the input s is a string.

```
def mystery(s):
    t = ""
    for c in s:
        if "A" <= c <= "Z":
            t += c
    return t
```

Answer:

#4 - Code Tracing with Strings - 10pts

Write below the following block of code the lines of text that it will print when run.

```
s1 = "15-110 is cool"
s2 = "CMU rocks!"
print(s1[5] + s2[7])
print(s1[len(s1)-1] + s2[1])
print(s1[4:8])
print(s2[2:len(s2)-2])
print(s1[::4])
```

Answer:

#5 - Linear Search Debugging - 10pts

The following three functions are all attempting to implement linear search, as we discussed in lecture. Only one is correct. Identify which of the three functions is correct, then explain what is wrong with the other two and how they can be fixed.

```
def linearSearchA(s, c):
    i = 0
    for char in s:
        i = i + 1
        if char == c:
            return i
    return -1

def linearSearchB(s, c):
    i = 0
    while i < len(s):
        if s[i] == c:
            return i
        i = i + 1
    return -1

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

Which implementation is correct?

- ❑ linearSearchA
- ❑ linearSearchB
- ❑ linearSearchC

Why are the other two incorrect? Answer:

#6 - File Representation - 5pts

In twenty words or less, why does the computer display random characters if you try to open a .png file in a text editor?

Answer:

#7 - Code Tracing with Input - 5pts

The following program asks the user for input multiple times. There is a possible sequence of inputs that leads to the program printing "SUCCESS". Write out that sequence of inputs below the block of code.

```
a = input("Let's play a game. What is my favorite number? ")
if int(a) == 42:
    b = input("That's right, well done! One more: what is 1/0? ")
    if int(b) == 1/0:
        print("SUCCESS")
    else:
        print("WRONG")
elif int(a) >= 40 and int(a) < 50:
    b = input("Close enough. Now, what word am I thinking of? ")
    if b != "Lovelace":
        print("WRONG")
    else:
        c = input("Good guess! Finally, what's 15 - 110? ")
        if int(c) == -95:
            print("SUCCESS")
        else:
            print("WRONG")
else:
    print("WRONG")
```

Answer:

Programming Problems

Each of these problems should be solved in the starter file available on the course website. They should be submitted on Gradescope under HW2 Full Assignment (Programming) to be autograded. Note that #5 and #7 will be manually graded.

Most programming problems may also be checked by running the starter file, which calls the function testAll() to run test cases.

#1 - pythagoreanChecker(a, b, c) - 5pts

Write the function pythagoreanChecker(a, b, c) which takes three integers, a, b, and c, and checks whether they are a Pythagorean triple. Three numbers (x,y,z) are a triple when x squared plus y squared is equal to z squared. Note that the numbers a, b, and c may be given in any order.

#2 - compoundInterest(base, rate, years) - 5pts

Write the function compoundInterest(base, rate, years), which calculates the amount that a base sum will have increased based on an annual interest rate and the number of years that have passed. This is computed by adding the current amount of money times the rate to the base sum every year.

For example, assume that you invest $1,000 in a retirement account that returns 0.25% interest annually in 2019. In 2020, the account will have $1,002.50; in 2021, the account will have ~$1,005.01; and by 2059, the account will have ~$1,105.03. Note that you do not just add $2.50 every year; the amount of interest increases as the base sum increases. You can read more about compound interest rates at https://en.wikipedia.org/wiki/Compound_interest

You may assume that years is an integer, base and rate are numbers, and that all three numbers are non-negative.

#3 - factorial(x) - 5pts

Write the function factorial(x) which takes a nonnegative integer, x, and returns x!. Recall that x! = x*(x-1)*(x-2)*...*3*2*1. **You may not use the built-in function math.factorial; that would make this too easy.**

#4 - countSentences(s) - 5pts

Write the function countSentences(s) that takes a string s and returns the number of non-empty sentences that occur in s. We define a sentence to be a consecutive string of one or more non-whitespace characters that ends in one of the following characters: . ! ?

For example, the following string has three sentences.

"You've got to ask yourself a question. Do I feel lucky? Well, do ya, punk?!"

Note that if a sentence ends in multiple punctuation marks, it still only counts as one sentence. Also note that the test cases are guaranteed to not use ., !, or ? inside a sentence, to simplify the problem.

Hint: consider using the string methods we discussed in class to make this problem much easier. Specifically, s.replace() and s.split() might be helpful...

#5 - printTriangle(n) - 10pts

Write a function printTriangle(n) which prints an ascii-art triangle out of asterisks based on the integer n (which is guaranteed to be positive and odd). For example, printTriangle(5) would print the following:

```
*
**
***
**
*
```

Note that the triangle is five lines long, with the top and bottom line each having only one asterisk, the second and second-from bottom line each having two asterisks, etc. So printTriangle(9) would look like:

```
*
**
***
****
*****
****
***
**
*
```

#6 - decodeFile(filename) - 10pts

Write the function decodeFile(filename) which takes a string filename as input, gets the text out of the file named filename, decodes a secret message in the text, and returns that secret message.

The file that the function is given will have a format like the one shown below:

72 101 108 108 111
87 111 114 108 100 33

To decode the message, you need to transform each number into its ascii string value, by using the chr() built-in method. If you combine together the resulting characters in the same line, you'll get a word. Combine together the words formed by every line (separated by spaces), and you'll get the secret message!

In the example shown above, the first line produces "Hello" and the second line produces "World!", so the function would return "Hello World!"

Make sure to download the files test1.txt and test2.txt and put them in the same directory as your hw2.py file, so you can test your code!

#7 - printPrimeFactors(x) - 10pts

Write the function printPrimeFactors(x) which takes a positive integer x and prints all of its prime factors.

A prime factor is a number that is both prime and evenly divides the original number (with no remainder). So the prime factors of 70 are 2, 5, and 7, because 2 * 5 * 7 = 70. Note that 10 is not a prime factor because it is not prime, and 3 is not a prime factor because it is not a factor of 70.

Prime factors can be repeated when the same factor divides the original number multiple times; for example, the prime factors of 12 are 2, 2, and 3, because 2 and 3 are both prime and 2 * 2 * 3 = 12. The prime factors of 16 are 2, 2, 2, and 2, because 2 * 2 * 2 * 2 = 16.

This problem is slightly more complex than most of the other homework problems, so let's lay out a high-level algorithm for how to approach it. First, note that when we solve this problem by hand, we can find the prime factors of a number by repeatedly **dividing the number** by the smallest possible factor until the number becomes 1. So our algorithm might look something like this:

1. Repeat the following procedure until the number x becomes 1
    a. Define a new number, n, to be 2
    b. Repeat the following procedure until a factor is found
        i. If the current number divides n evenly
            1. Print the number n
            2. Set x to x divided by n
            3. Report that a factor has been found
        ii. If it does not
            1. Add one to n