COM S 1270 Exam #2

Spring 2025

Name: _____ **Student ID #: _____**

ISU username/ netid: _____**@iastate.edu**

General Instructions:

- Please **look over the exam carefully** before you begin.
- **READ ALL OF THE INSTRUCTIONS CAREFULLY – THIS IS THE POINT OF THE EXAM.**
- **NO, REALLY, READ ALL OF THE INSTRUCTIONS VERY CAREFULLY.**
- The problems are **not** necessarily **in order of difficulty**.
- **Closed book/ notes, Closed internet/ email, Closed friend/ talking**.
- **NO ELECTRONIC DEVICES/ NO HEADPHONES**.
- **Time Limit**: 75 minutes.
- **Use correct Python syntax** for writing any code – including the use of whitespace.
- If you like, you may draw vertical straight lines to denote different levels of whitespace for clarity.
- You are **not required** to **write comments** for your code. However, **brief comments may help make your intentions clear if your code is incorrect**. **DO NOT WRITE ANY CODE NOT ASKED FOR IN THE QUESTION.**
- There are five (5) questions on the exam.
- **Mark the three (3) questions you wish to have graded with a star on the question number** (★).
- If you do *not* mark three (3) questions with a star (★), questions will be graded starting at question A.
- The questions you select to be graded will be worth thirty (30) points each.
- You **must** at least *attempt* the other two (2) questions.
- Regardless of the correctness of the answers for the two (2) questions you do not select to be graded, so long as you *try* you will receive five (5) points each.
    - Here, 'trying' means providing code or comments explaining the full solution. Meaning – a partial solution is insufficient for the purposes of 'trying.'
- If you do *not* attempt one or more of the other two (2) non-graded questions, you will *not* receive the points for each question you do not attempt.

Lab Day: _____ Lab Start Time: _____

Question	Student's Score	Max Score
GR #1:		30
GR #2:		30
GR #3:		30
AT #1:		5
AT #2:		5
TOTAL:		100

Excerpts from the Python API documentation

Built-in functions	
`print(x)`	Prints value to console with newline.
`print(x, end='')`	Prints value to console without newline
`int(x)`	Convert to type int
`float(x)`	Convert to type float
`str(x)`	Convert to type string
`len(x)`	Length of string or list
`abs(x)`	Absolute value of a number
`round(x)`	Rounded value of a number
`pos(x, y)`	x to the power y
`math.sqrt(x)`	square root of x (requires `import math`)
`ord(s)`	character code for a character (single-character string)
`chr(x)`	character whose code is the number x
`range(start, bound)`	range of numbers from start up to (but not including bound)
`range(start, bound, step)`	range of numbers from start up to (but not including bound), in increments of step
`max(n1, n2, ...)`	maximum of a bunch of numbers
`min(n1, n2, ...)`	minimum of a bunch of numbers
String methods	
`s.upper()`	returns an uppercase version of string s
`s.lower()`	returns a lowercase version of string s
`s.title()`	returns a titlecase version of string s (all words capitalized)
`s.strip()`	returns a string obtained from s by removing all leading and trailing whitespace
`s.replace(oldstring, newstring)`	returns a string obtained from s by replacing each occurence of oldstring with newstring
`s.startswith(t)`	returns True if the beginning of string s matches string t
`s.endswith(t)`	returns True if the end of string s matches string t
`s.isdigit()`	returns True if all characters of s are numeric digits
`t in s`	expression is True if t is a substring of s
`s.count(t)`	returns the number of times t occurs as a substring of s

`s.find(t)`	returns the index of the first occurrence of t as a substring of s (-1 if t isn't a substring)
`s.find(t, start)`	returns the index of the first occurrence of t as a substring of s, beginning the search at position start (-1 if t isn't a substring)
`s.rfind(t)`	returns the index of the last occurrence of t as a substring of s (-1 if t isn't a substring)
`s.split()`	returns a list of substrings of s delimited by whitespace
`s.split(t)`	returns a list of substrings of s delimited by occurrences of string t
`s.join(lst)`	returns a a string obtained by concatenating together the strings in the list lst, with the string s between them
List methods	
`lst.append(x)`	appends item x to the end of list lst
`lst.insert(pos, x)`	inserts item x at the given index pos
`lst.pop()`	removes and returns the last item in the list
`lst.pop(pos)`	removes and returns the item at index pop
`lst.remove(x)`	removes the first occurrence of x in the list
`lst.count(x)`	counts the number of occurrences of x in lst
`lst.index(x)`	returns the index of the first occurrence of x in the list (error if x is not in the list)
`lst.sort()`	sorts the elements of lst
`lst.reverse()`	reverses the elements of lst
`lst.clear()`	removes all the elements of lst
`lst.extend(lst2)`	appends all elements of lst2 onto lst

A) Consider the given function, **isPrime()**, that determines whether the given integer, **n**, is prime (greater than 1 and divisible only by 1 and itself).

E.g., 7 is prime, 6 is not prime.

Using the given **isPrime()** function, write a function, **listPrimes()**, that returns a list of the first **howmany** prime numbers. The function returns an empty list when the value of **howmany** is less than or equal to zero.

E.g., **listPrimes(5)** returns [2, 3, 5, 7, 11] and **listPrimes(-2)** returns [].

```
def isPrime(n):
    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

def listPrimes(howmany):
```

Syntax: _____ + Logic: _____ + Output: _____ = Total: _____

B) Write a function called **transformString()** that, given a string, **s**, returns a new string where any doubled character in **s** is tripled in the new string. Meaning, any character in **s** that appears twice in a row should now appear three times in a row in the new string. Additionally, any character that is already tripled in **s**, meaning it appears three times in a row, is replaced with "X" in the new string.

E.g.:

transformString("banana") returns "banana"
transformString("apple") returns "appple"
transformString("TTAT") returns "TTTAT"
transformString("AAA-AAA-AAA") returns "X-X-X"
transformString("Wooow!") returns "WXw!"

```
def transformString(s):
```

Syntax: _____ + Logic: _____ + Output: _____ = Total: _____

C) Consider the variables below:

```
x = 128
y = "programming"
```

Using the variables **x** and **y** above, for each phrase below, write a Python Boolean expression that captures its meaning. Assign the evaluation of that expression to a new variable called **z**. Then, determine whether the expression is **True** or **False** using the values of **x** and **y** above and place that information in a comment below your expression.

```
x is greater than 100
```

```
y contains the substring "gram"
```

```
x is divisible by 8 and x is not divisible by 7
```

```
the length of y is greater than or equal to 5
```

```
x is a multiple of the length of y
```

```
y starts with "p" and ends with "g"
```

Syntax: _____ + Logic: _____ + Output: _____ = Total: _____

D) Consider the following function:

```
def foo(a, b, c):
    result = False
    if a > b and c != 5:
        result = True
    if a == 10 or b % 2 == 0:
        result = True
    if a + b == c:
        result = False
    return result
```

Rewrite the function **foo()**, above, so that it returns the same results but does not contain any conditional statements (no if statements). Assume **a**, **b**, and **c** are integers.

```
def foo(a, b, c):
```

Syntax: _____ + Logic: _____ + Output: _____ = Total: _____

E) The Promo-Goods company sells custom-made t-shirts and mugs online. T-shirts are $15.00 each, and mugs are $5.00 each, but you get 1 free mug for every 3 t-shirts ordered. There is a service charge of $30 for orders of fewer than 50 items. Additionally, there is a shipping cost of $20 for all orders. Sales tax is added to the total order cost (excluding shipping but including the service charge, if applicable).

Write a function, **calculateTotal()**, which takes the numbers of t-shirts, mugs, and the tax rate as parameters, and which returns the total cost of a given number of t-shirts and mugs including tax and shipping. Assume the tax rate is given as a decimal. E.g., "8%" would be given as 0.08. This function does not read input or print anything.

E.g.: The local Art Club orders 9 t-shirts and 10 mugs with a tax rate of 8%. The t-shirts cost $135, they get 3 mugs free, and the remaining 7 mugs cost $35. Since fewer than 50 items were ordered, there is a service charge of $30. The subtotal before tax and shipping is $200. After adding the tax ($16 = $200 * 0.08) and shipping ($20), the order total is $236. Please note that any free mugs received do not impact the total number of items ordered. Meaning, if you order 49 t-shirts, you would get 16 mugs, but the service charge would still apply.

```
def calculateTotal(tshirts, mugs, tax):
```

Syntax: _____ + Logic: _____ + Output: _____ = Total: _____

Scratch Paper

Scratch Paper