COM S 1270 Exam #1

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) Transforming robot toys are $14.99 each or $69.99 for a 5-pack. Additionally, a discount is applied to orders of 10 toys or more: 5% off the total cost (excluding shipping and not including any tax). Sales tax is added to the total order cost (excluding shipping and after the discount, if applicable). Finally, shipping for the order is $7.00 per 5-pack or $2.00 for an individual toy.

Write a Python function called **transformingToysCost()** that takes the number of toys and the tax rate as parameters, and returns the total cost of an order of toys. Assume that the specific model of toy does not matter, and that you are only attempting to minimize the cost for your order. Assume the tax rate is given as a decimal. E.g., "8%" would be given as 0.08.

```
def transformingToysCost(number, tax):
```

Syntax: _______ + Logic: _______ + Output: _______ = Total: _______

B) Write a Python function called **nextSpecialMultiple()** that takes two parameters: **number** and **mult**. The function should return the smallest number that is strictly greater than the value provided by **number**, and which is also a multiple of **mult**. If either **number** or **mult** are less than or equal to zero, the function returns -1. E.g., **nextSpecialMultiple(22, 5)** returns 25, **nextSpecialMultiple(21, 7)** returns 28, and **nextSpecialMultiple(-2, 0)** returns -1.

```
def nextSpecialMultiple(number, mult):
```

Syntax: _______ + Logic: _______ + Output: _______ = Total: _______

C) Consider the function **levelAssessment()** below:

```
def levelAssessment(points):
    if points < 25:
        level = "Novice"
    elif points >= 25 and points < 50:
        level = "Beginner"
    elif points >= 50 and points < 75:
        level = "Intermediate"
    elif points >= 75 and points < 100:
        level = "Expert"
    elif points >= 100:
        level = "Master"
    return level
```

Modify this function so that it no longer uses the **and** operator. Additionally, ensure that the input value, **points**, is non-negative by using a 'guard clause' (an **if** statement that ensures correct input) at the top of the function. If the value of **points** is less than zero, immediately return the string **"ERROR"**.

```
def levelAssessment(points):
```

Syntax: _______ + Logic: _______ + Output: _______ = Total: _______

D) Write a Python function named **replaceCharWithCAT()** that takes two strings, called **text** and **char**, as parameters. The **text** parameter can consist of any number of characters, but if the **char** parameter has a cardinality greater than one or exactly equal to zero, the function returns the string “ERROR”. Otherwise, this function returns a new string consisting of the **text** string with any instances of **char** replaced with the string “CAT!”. E.g. **replaceCharWithCAT (“APPLE CAT!”, “A”)** returns “CAT!PPLE CCAT!T!”, **replaceCharWithCAT (“APPLE CAT!”, “Q”)** returns “APPLE CAT!”, and **replaceCharWithCAT (“APPLE CAT!”, “APP”)** returns “ERROR”.

NOTE: You may not use any built-in string methods to accomplish this task.

```
def replaceCharWithCat(text, char):
```

Syntax: _______ + Logic: _______ + Output: _______ = Total: _______

E) The Lucas sequence begins with 2 and 1. All subsequent values are calculated using the formula:

$$L(n) = L(n-1) + L(n-2)$$

E.g., 2, 1, 3, 4, 7, 11, 18, 29, etc.

Meaning, for $L(6) = 18 \implies 18 = 11 + 7$

Write a Python function, called **lucasString()**, which takes a zero-based index value, **n**, as a parameter. This function returns a string of the sequence up to and including the value at index **n**. Each sequence value in the string should be separated by a comma and a space. There should not be any commas or spaces at the end of the string. Any negative value for **n** should return the string "ERROR".

E.g. **lucasString(4)** returns "2, 1, 3, 4, 7", while **lucasString(-2)** returns "ERROR".

```
def lucasString(n):
```

Syntax: _______ + Logic: _______ + Output: _______ = Total: _______

Scratch Paper

Scratch Paper