# CSE 1321L: Programming and Problem Solving I Lab

## Assignment 3

## Module 2 Part 2

## What students will learn

- Problem Solving.
- Basic Program Structure.
- Input and Output with the user.
- Write code that includes while/for loop and nested loops logic.
- Structure program to include conditional logic.

## Content



## Overview:

For this assignment, you're going to practice making logic in your code. It will include loops and nested loops. In practical terms, this means you're going to expand on the concepts from previous assignments, but also include things like while loop, for loop statements. Again, start early, practice, and ask a lot of questions.

**Final note: *Do not cheat***

If your temptation is to look online, don't. Come see us instead and ask questions – we are here to help. Remember, you are going to have to write codes in your future job interviews, so learn it now to secure a high-paying job later.

## Assignment3A: Number Diamond

This program generates a diamond pattern of numbers using nested loops. The shape will be composed of single digit numbers 0-9. The size of the diamond will be determined by a single value inputted by the user.

**Requirements:**

- Your solution must use a nested FOR loop exclusively.
- Assume the user will input any positive value for the input.
- The input size will determine the diamond's height and width.
- For this shape to work, we can only use odd values for size. If the user enters an even number, the program will automatically increase it by 1.
- Use the Modulus % operator to figure out if the value is even or odd.
- The diamond should be filled with increasing single digit numbers starting from 0 to 9. Once the number gets to 9, we will re-start from 0 and so on. Use the Modulus % operator for this.
- As always, you must follow the output format provided in the Sample Outputs

**Hints:**

- You may have to heavily rely on Modulus for this lab so make sure you understand how to use them.
- Count how many numbers and how many "left side empty spaces" there are on each line and see if you notice a pattern.

**Sample Output #1:**

```
Enter an odd number for the size of the diamond: 7
   0
  123
 45678
9012345
 67890
  123
   4
```

**Sample Output #2:**

```
Enter an odd number for the size of the diamond: 8
Size must be an odd number; we will increase it to 9
    0
   123
  45678
 9012345
678901234
 5678901
  23456
   789
    0
```

**Sample Output #3:**

```
Enter an odd number for the size of the diamond: 12
Size must be an odd number; we will increase it to 13
      0
     123
    45678
   9012345
  678901234
 56789012345
6789012345678
 90123456789
  012345678
   9012345
    67890
     123
      4
```

## Assignment3B: Count Character Frequency

Write a Python program that takes a string as input and counts how many times each **character** appears, **ignoring spaces**. Once a character's frequency has been counted, replace it with a number to avoid recounting it.

**For this task:**

- You **cannot** use lists or dictionaries to store frequencies as we have not covered yet.
- You **cannot** use the len() function. Iterate on string sequence
- You are allowed to use the replace () function to modify the string.
  - The replace () function in Python is used to replace parts of a string with a new substring. The syntax is: string.replace(old, new).
    - old: the char you want to replace.
    - new: the char you want to replace it with.

**Note:**

- Space is also a character, and we don't want to count the frequency of space. You can think about getting rid of spaces by using replace function described above.
- Take care of upper and lower cases.

Example runs are shown below. The user input is shown in **red and bold** **(Notice the difference between time & times)**

**Sample Output #1:**

```
[Character Frequencies]
Enter a string: Apple Banana
a appears 4 times
p appears 2 times
l appears 1 time
e appears 1 time
b appears 1 time
n appears 2 times
```

**Sample Output 2:**

```
[Character Frequencies]
Enter a string: hello worlddd
h appears 1 time
e appears 1 time
l appears 3 times
o appears 2 times
w appears 1 time
r appears 1 time
d appears 3 times
```

**Sample Output 2:**

```
[Character Frequencies]
Enter a string: Aaa
a appears 3 times
```

## Assignment3C: Guess the Word Game

Write a Python program that plays an interactive **"Guess the Word"** game with the user. In this game, the user will try to guess a word by guessing one letter at a time. The program will display the word with some characters hidden as underscores (_). When the user guesses a correct letter, the corresponding underscores will be replaced by that letter. The game will continue until the user guesses all the letters correctly.

You **cannot** use lists or arrays, but you can use the len() function. You should only use strings and simple loops. The program must be able to interact with the user by taking input and displaying results after each guess.

**For this task:**

1. **Initial Input:**
   - The program should first prompt the user to input a word to guess. This word should be kept hidden from the user and will only contain lowercase letters.
2. **Word Display:**
   - Initially, display the word as a string of underscores (_), with each underscore representing a letter in the word. For example, if the word is "apple", display it as _ _ _ _ _.
3. **Guessing:**
   - The program should repeatedly ask the user to guess one letter at a time.
   - The user will input a letter, and the program should check if that letter exists in the word.
4. **Correct Guess:**
   - If the guessed letter is correct (i.e., it exists in the word), replace the corresponding underscores with the guessed letter.
   - For example, if the word is "apple" and the user guesses "a", the display should change to **a** _ _ _ _.
5. **Incorrect Guess:**
   - If the guessed letter is incorrect (i.e., it does not exist in the word), the program should notify the user that the guess is wrong and prompt them to try again.
6. **Game End:**
   - The game continues until all the letters in the word have been guessed correctly, replacing all underscores with the correct letters.
   - When the word is completely guessed, the program should display a congratulatory message and end the game.

**String Indexing:**

In Python, you can access individual characters of a string using indexing. Each character in a string has a unique position, starting from 0.

- For example, in the word "apple", the index of the letter "a" is 0, the letter "p" is 1, and so on.
- You can use string[i] to access the character at position i in a string.
- For example, if you want to get the first letter of the word "apple", you would use word[0], which will give "a".
- Similarly, word[1] will give "p", word[2] will give "p", and so on.

**Note**

In this problem, you'll need to use string indexing to compare each character in the word to the guessed letter and update the displayed word accordingly.

Example runs are shown below. The user input is shown in **red and bold.**

**Sample Output #1:**

```
Welcome to the Guess the Word game!
Enter a word to guess (lowercase letters only): apple
The word to guess is: _ _ _ _ _
Guess a letter: q
Oops! That letter is not in the word.
The word to guess is: _ _ _ _ _
Guess a letter: a
Good guess!
The word to guess is: a _ _ _ _
Guess a letter: r
Oops! That letter is not in the word.
The word to guess is: a _ _ _ _
Guess a letter: p
Good guess!
The word to guess is: a p p _ _
Guess a letter: l
Good guess!
The word to guess is: a p p l _
Guess a letter: e
Good guess!
Congratulations! You've guessed the word: apple
```

**Sample Output 2:**

```
Welcome to the Guess the Word game!
Enter a word to guess (lowercase letters only): python
The word to guess is: _ _ _ _ _ _
Guess a letter: p
Good guess!
The word to guess is: p _ _ _ _ _
Guess a letter: y
Good guess!
The word to guess is: p y _ _ _ _
Guess a letter: t
Good guess!
The word to guess is: p y t _ _ _
Guess a letter: h
Good guess!
The word to guess is: p y t h _ _
Guess a letter: o
Good guess!
The word to guess is: p y t h o _
Guess a letter: n
Good guess!
Congratulations! You've guessed the word: python
```

## Submission Instructions:

- Programs must follow the output format provided. This includes each blank line, colons (:), and other symbols.
- Programs must be working correctly.
- Programs must be written in Python.
- Programs must be submitted with the correct **.py** format.
- Programs must be saved in files with the correct file name:
  - Assignment3A.py
  - Assignment3B.py
  - Assignment3C.py
- Programs (source code files) must be uploaded to Gradescope by the due date.