CSE 1321L: Programming and Problem Solving I Lab

Assignment 6 – 100 points

Lists

What students will learn:

1) Declaring lists

2) Performing common operations on lists

Overview: Lists are an incredibly powerful thing in computing. Almost every audio file, video file,and image you've ever seen on a computer is stored in a list-like structure. A list is simply adata structure that holds a lot of values. For example, they could hold 50 integers, 100 Booleans, or a million floats.

map() : The map() function in Python is used to apply a given function to every item in an iterable (like a list or a string) and return a map object (which can be converted into a list).

Example:

```
# Convert a list of string numbers into integers

numbers = ['1', '2', '3', '4']

integer_numbers = list(map(int, numbers))  # Applies int() to each element in the list

print(integer_numbers)
```

Output:

```
[1, 2, 3, 4]
```

split() : The split() function is typically used on strings. It splits a string into a list where each word (or part of the string) is separated by a space by default (you can also specify a different delimiter).

Example:

```
# Split a string by spaces

sentence = "Hello world Python"

words = sentence.split()  # Splits by space by default

print(words)
```

Output:

```
['Hello', 'world', 'Python']
```

sum() : The sum() function is used to add up all the elements in an iterable (like a list).

Example:

```
# Sum up all numbers in a list
numbers = [1, 2, 3, 4]
total = sum(numbers)
print(total)
```

Output:

```
10
```

Assignment 6A: Sorting with Bubble Sort

Write a Python function called bubble_sort which takes in a list of numbers. It must perform a bubble sort on the list. Note you may not use the built in list.sort function, you must implement a bubble sort from scratch. It must return a sorted list of numbers.

Write a Python function called count_positive which takes in a list of numbers. It must count how many numbers in the list are positive. Return the number of positive numbers in the list.

Write a Python function called count_negative which takes in a list of numbers. It must count how many numbers in the list are negative. Return the number of negative numbers in the list.

Outside of the above functions, prompt the user to enter in a line of numbers. The users should give you a string made up of numbers separated by spaces. Convert the string into a list of numbers.

Call the bubble_sort function passing it the list of numbers you created from the users input.

Call the count_positive, and count_negative functions, and store the result.

Finally, print out the sorted list, and a count of how many numbers are positive and how many are negative.

Requirements:

- You must implement the bubble sort algorithm manually.
- Do not use any built-in sorting functions.

Sample Input:

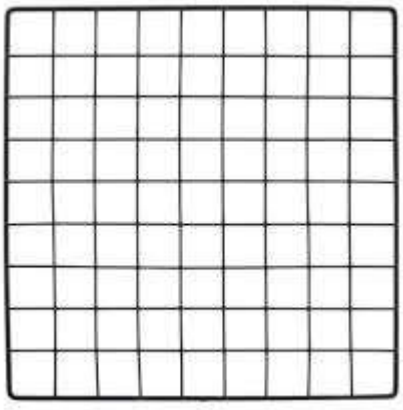
```
Enter numbers separated by space: 3 -1 0 5 -7 8
```

Sample Output:

```
Sorted List: [-7, -1, 0, 3, 5, 8]
Positive Numbers: 3
Negative Numbers: 2
```

Assignment 6B : Dungeon Treasure Map

In this assignment you will create a treasure hunt board game. Players will play on a board which resembles a grid like this:



The exact dimensions of the grid will be determined by the player.

Each square of the grid will either have an 'O' representing an open square, a 'T' representing treasure, or an 'X' representing treasure that has been collected.

Steps:

1) Prompt the user for the size of the grid (width and height)
2) Create a list called board, each cell will represent a row of the grid.
3) In the first cell of the board list, you'll place a list.

    a. Pick a random number between 0 and 1. If the number is greater than or equal to 0.7 you'll add a Treasure 'T' to the next cell of the list. If the number is less than 0.7 you'll add an open 'O' to the next cell of the list.
    b. Keep track of how many treasures you are adding to the board in a separate variable called numberOfUndiscoveredTreasures.
    c. Repeat step (a) until you have a list that is the height the user asked for in step (1).
4) Repeat step (3) until the board is the width the user asked for in step 1.
5) Tell the user how many treasures you have hidden.
6) Next you'll ask the user to guess coordinates, you'll check if they found treasure or not:
    a. Ask the user to enter in a row number (0 to the width of the board -1)
    b. Ask the user to enter in a column number (0 to the height of the board -1)
    c. Check that location to see if it is a "T" (Treasure) or an "O" (Open).
        i. If it's a treasure tell the user they got treasure, change that cell of the board to an "X" to indicate that it was already discovered. Lower the number of undiscovered treasures by one.
        ii. If it's not a treasure, tell the user to try again.
        iii. Keep asking the user to guess locations until the user has discovered all the Treasures, then print out the whole board, and end the game.

Sample Input:

```
Enter dungeon width: 4
Enter dungeon height: 4
Treasures are hidden in 3 locations.


Enter row to check (0-3): 2
Enter column to check (0-3): 3
You found a treasure at (2, 3)!
Enter row to check (0-3): 1
Enter column to check (0-3): 1
No treasure found at(1, 1)
Enter row to check (0-3): 1
Enter column to check (0-3): 2
```

```
You found a treasure at (1, 2)!
```

Sample Output:

```
O O O O

O O X O

O O O X

O O O O
```