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 a data structure that holds a lot of values. For example, they could hold 50 integers, 100 Booleans, or a million floats.

Assignment 6A: *Keeping Score.* Now that we know about lists, we can keep track of **when** events occur during our program. To prove this, let's create a game of Rock-Paper-Scissors. For those not familiar with this game, the basic premise is that each player says "Rock-Paper-Scissors!" and then makes their hand into the shape of one of the three objects. Winners are determined based on the following rules:

Rock beats Scissors
Scissors beats Paper
Paper beats Rock

At the start of the program, it will ask the player how many rounds of Rock-Paper-Scissors they want to play. After this, the game will loop for that many number of times. Each loop, it will ask the player what item they want to use – Rock, Paper, or Scissors. The computer will **randomly generate** its own item, and a winner will be determined. The game will then save the result as an element of a list, and the next round will begin. Once all the rounds have been played, the program will say "Game Over" and display a list of who won each round, in order.

Hints: While the random functions we've discussed thus far can't generate words, we can associate numbers with them instead. For example, we might treat the computer randomly generating 0 as if it threw Rock. We could then use that number to access a predefined string list with "Rock", "Paper", and "Scissors" stored inside it.

Sample Output #1:

```
How many games do you want to play?: 3
Round 1: What do you want to throw?: Rock
Computer threw SCISSORS!
Round 2: What do you want to throw?: Paper
Computer threw PAPER!
Round 3: What do you want to throw?: Rock
Computer threw PAPER!
Game Over...
```

```
Here’s the recap:
Player won Round 1 with Rock
Tied on Round 2 with Paper
Computer won Round 3 with Paper
```

Sample Output #2:

```
How many games do you want to play?: 2
Round 1: What do you want to throw?: Scissors
Computer threw ROCK!
Round 2: What do you want to throw?: Paper
Computer threw SCISSORS!
Game Over...

Here’s the recap:
Computer won Round 1 with Rock
Computer won Round 2 with Scissors
```

Assignment 6B: *Level Map Creator.* There are a variety of ways that game developers store their level layouts. One simple method is to associate level elements with certain symbols, and then storing them in a 2D grid inside a text file. We will use our knowledge of 2D lists to create a very simple Level Map Creator tool.

The program should prompt the user to enter a width and height for the level. Then it should initialize a 2D list and fill every element with the "**_**" symbol. Afterwards, the user should be given the following options via a menu:

1. Clear Level	Re-initialize the 2D list and fill every element with the "**_**" symbol.
2. Add Platform	Prompt the user to enter a starting point and length for the horizontal platform. Replace those elements in the 2D list with the "**=**" symbol. If the length is longer than the number of columns (or out of bounds), notify the user that this is not possible.
3. Add Item	Prompt the user to enter a column and row index. Replace that element in the 2D list with the "**P**" symbol. If the column and row index in outside the bounds of the list, notify the user that this is not possible.
4. Quit:	End the program

After completing the task, print the modified 2D list, If anything other than Quit is selected, display the menu again. Otherwise, tell the player "Good bye!" and stop.

Hints: Since we’re using a 2D list to represent the level map, we’ll use its indexes for our level coordinates. 0,0 will be the top-left corner of the map, and will correspond to list[0][0].

Sample Output #1:

```
[FYE Level Map Creator]
Enter a level map width: 20
Enter a level map height: 6
____________________
____________________
____________________
____________________
____________________
____________________

Options
1. Clear Level
2. Add Platform
3. Add Items
4. Quit
Enter a choice: 2

[Add Platform]
Enter X Coordinate: 1
Enter Y Coordinate: 1
Enter Length: 5
____________________
_=====______________
____________________
____________________
____________________
____________________

Options
1. Clear Level
2. Add Platform
3. Add Items
4. Quit
Enter a choice: 3

[Add Item]
Enter X Coordinate: 30
Enter Y Coordinate: 30
This is not a valid location!
____________________
_=====______________
____________________
____________________
____________________
____________________

Options
1. Clear Level
2. Add Platform
3. Add Items
4. Quit
Enter a choice: 3

[Add Item]
Enter X Coordinate: 3
Enter Y Coordinate: 0
___P________________
_=====______________
____________________
```

```
____________________
____________________
____________________

Options
1. Clear Level
2. Add Platform
3. Add Items
4. Quit
Enter a choice: 2

[Add Platform]
Enter X Coordinate: 3
Enter Y Coordinate: 1
Enter Length: 29
This platform won’t fit in the level!
___P________________
_=====______________
____________________
____________________
____________________
____________________
Options
1. Clear Level
2. Add Platform
3. Add Items
4. Quit
Enter a choice: 1

[Clear Level]
____________________
____________________
____________________
____________________
____________________
____________________

Options
1. Clear Level
2. Add Platform
3. Add Items
4. Quit
Enter a choice: 4

____________________
____________________
____________________
____________________
____________________
____________________

Goodbye!
```

Submission:

1. You will submit 2 separate files
2. File names must be correct.
3. Upload all files (simultaneously) to the assignment submission folder in Gradescope.