Due Date: 11/11/24 @ 11:59 PM

CSC240 Assignment #10

This week, you guessed it, we are learning a new data structure, DS_Stacks! Stacks are a fantastic data structure for scenarios where we need to maintain the order of our elements, but only need to access the last element that was inserted. Stacks follow a Last In, First Out (LIFO) order, meaning the last element inserted is the first element that can be removed (think of a stack of plates, the last plate you wash & dry and place in the cupboard is the first one you'll pick up and use). Stacks have two primary operations: push(), which adds an element on top of the stack, and pop(), which removes the top element from the stack and returns it. One other common operation with Stacks is peek() (called top() in GameMaker cause they always have to try and be unique 🙄), which returns the element on top of the stack, without actually removing the element from the stack. Of course, other basic operations exist for checking the number of elements on the stack (size) or if it is empty (empty), but these are the main methods utilized.

This week, we are going to utilize DS_Stacks to implement a childhood classic card game, Crazy Eights! This implementation will follow the rules found here: https://bicyclecards.com/how-to-play/crazy-eights (I implemented a score limit of 100, since the page does not provide a standard score limit). With just a few Structs, Stacks are an excellent structure for assisting us in representing a deck of cards and various play piles in many different card games (only one play pile in Crazy Eights though)!

For this assignment, just like last time, I have created the UI, Gamemaker Objects, and defined a few Structs that are used in the implementation. You are responsible for creating the Deck and Pile (CrazyEightPile) structs, as they will utilize a Stack of Cards in the implementation. The script files containing the structure of the Deck and Pile Structs are already created for you. Inside of these Structs, you will find that I have provided for you the following:

- All attributes as well as brief descriptions of what each of these attributes are used to represent/control
- The names and parameters of all of the methods that you must implement. (NOTE: function names must remain the same, as they are referenced in various areas of the code)
- Documentation for each function that specifies what that function is to do, what the parameters represent, and what the return value should represent.

You are simply responsible for filling in the empty Struct methods with the code that would perform the specified actions. This time, I have further modularized our code by implementing Structs for each of the User Interface elements. These Structs are preceded by "ui". All other Structs are abstract implementations for storing the Game state (aka the Game Model), as in past projects. I would suggest porting these Model Structs (mainly the Card, Deck, Hand, and Pile structs) out into a blank project and testing each of the functions as you write them. You will also notice that this time, the .toString() methods for Deck and Pile are left empty. You are responsible for writing these .toString() methods for the Deck and Pile Structs (.toString() for Hand, Card, and Game are still provided to you). Once you believe that you have the Deck and Pile Structs up and running, paste this into the fully-working project that I have provided to you and if it works properly, then congratulations, you've done it!

Due Date: 11/11/24 @ 11:59 PM

Please note: You only need to edit the Deck and Pile Struct script files. There does not need to be any other changes made to any of the other files or code in the Gamemaker objects, however, I would highly recommend taking a look at these areas and understanding what they do, as this may make your implementation a bit easier, as well as thoroughly prepare you for designing your Final Project. I have made sure to incorporate thorough documentation to guide you through all areas of the program. 😊

When you have completed the assignment, please zip up your Game folder and submit it to the Week 10 Assignment Dropbox.