Programming Assignment #6

General

This project will give you a chance to write some recursive functions. **Please Note: Your functions should not use any loop or global variables unless specifically allowed in the function's instructions.**

Your Mission

Edit the file `Project6.cpp`. This project is sort of like a game where you are trying to complete various games in order to find out how the story ends. After every completed function, you will progress in the story and eventually find out what happens. So exciting!

Chapter 0: Tutorial

To get your toes a little wet, we are going to write two flavors of a recursive sum of an array without any loops. These functions will not be graded and are merely for your practice. However, you may find the solutions useful when doing other parts of the lab. Note that we also check the number of function calls for you. This feature is not implemented for the other methods but will be tested and graded. Feel free to implement the checks yourself.

Chapter 1: The Enigma of the Reversed Number

Write a function to reverse a `n` digit number using only recursion. `reverse` will work based on a recursive approach known as divide-and-conquer where you split the problem into smaller sub-problems and then assemble the pieces at the end. For example: `int` ”123” should become ”321” after the function has reached its end. Hint: You may use the pow() function from math.h. Also, you do not need to worry about appending leading zeroes (reverse(100) should just output 1, not 001).

Chapter 2: The Legend of the Linked Ladder

Write a function to recursively remove `nodes` of a `linked list` if they contain a specific value. The linked list node struct has been provided for you. Assume all nodes are allocated memory on the heap, so When you remove a node, be sure to free it.

Chapter 3: The Weighty Conundrum at the Elemental Fountain

Write a function to find the maximum weight that a 'knapsack' can hold. You will be given a list of items that all have unique weights associated with them as well as a maximum weight that the 'knapsack' can hold. We want you to find the maximum weight that can be put in the bag without it ripping. There will be times when the maximum weight of the bag will not be met based on the items you are given. This is known as a ”toggle” recursion problem where you will try ”toggling” different item combinations in order to find the ideal combination.

Chapter 4: The Labyrinth of Infinite Forks

Write a function to find the shortest path through a maze of forks. The 'forks' are organized in a data structure called a binary tree. A binary tree is a series of `nodes` which are connected together, each with at most two children. This is when a specific 'door' is connected to a fork with two possible 'door' routes via a left and right child node pointer. You will be given a starting 'door' and should navigate through the tree and return the shortest distance from the start to an exit. An exit is reached when one of the doors leads to nowhere (a NULL pointer). Keep in mind this binary tree does not have to be 'balanced'- meaning a `node` may only have a single left or right child.

Part 5: The Lost Chapter

Write a function that calculates all of the possible wins and draws for a given tic-tac-toe board. You are provided a `game` struct that has the number of wins for 'X' and 'O' as well as the number of draws. This is a classic backtracking problem where you try every possible combination of 'X' and 'O'. You will also be responsible for checking if the current board state already has a winner or is a draw (i.e. if we give you a full board with no winner, you should return that the current board state has 1 possible draw). We HIGHLY recommend writing a helper function for this "win check" so that your function doesn't get unnecessarily long. For this function, we have also opted to not provide given/visible test cases. It is up to you to make these yourself.

CHECKLIST – Did you remember to:

- ☐ Re-read the requirements after you finished your program to ensure that you meet all of them?
- ☐ Make sure that your program passes all our test cases?
- ☐ Make up your own test cases?

Submission

Only submit your `Project6.cpp` file to GradeScope.