Data Structures

Assignment 2

(1) (15 pts) Write a C++ function which takes an array of int as a parameter and returns the largest element in the array. Pass the array using a pointer and remember the name of the array automatically converts to a pointer to the first element of the array. Note that, unlike vectors and strings, arrays do not carry their size with them. So, you'll have to also pass an int parameter that tells the function how many elements are in the array. In the function, iterate through the array using a pointer. Do not index it. Access the array elements by dereferencing the pointer.

(2) (25 pts) Write a C++ class that represents a combination lock. It should have a three-digit combination that is set in the constructor. The device will be unlocked by turning left a number of ticks equal to the first digit, then right a number of ticks equal to the second digit, and then left again. Any deviation from this sequence will not unlock the lock and the lock will then need to be reset in order to make any further progress towards unlocking.

There should be turn_left() and turn_right() functions that take a number of ticks as a parameter and represent the action of turning the lock left or right as the user tries to open it. An open() function should return "Open" if the lock is able to be opened and "Locked" if the lock is still locked. (Trying to open it when it's still locked does not force the lock to need a reset before further progress can be made.)

There should be a reset() function that lets the user try to enter the correct combination again after a failed attempt.

Declare an object of type Lock in your main() function and demonstrate your functions work as they should.

(3) (60 pts) Write a C++ program that plays the game of Nim. In Nim, there is a pile of tokens and each player takes turns removing tokens. A player must take at least one token but cannot remove more than half the tokens in one turn and the player who removes the last token loses the game.

Decompose the problem into three classes: Player, Pile, and Game. Your main function should declare an object of type Game and call the .play() function of game.

The Player class should consist of a player name and have a make_move() function that takes a Pile as a parameter and generates a legal move. If the Player name is "Computer" then this move should be random. If the Player name is anything else, then the move should be input from the keyboard.

The Pile class should contain a number of tokens, a max_legal_move() function that returns the maximum legal move, a remove_tokens() function that takes a number of tokens as a parameter and updates the pile accordingly. The constructor should initialize the number of tokens randomly between 50 to 150.

The Game class should contain two Players and a Pile. Its play() function should run through the gameplay. For each move, output the Player names and how many tokens are left in the Pile. Give appropriate feedback to the user about the game state, prompt for move, announce the winner, etc. The Game constructor should take two player names.

In your class design, incorporate principles of encapsulation. Do not expose data members to other classes. They should be accessed through apprpriate get() and set() methods. Write appropriate constructors for each class.