Homework #3 - Stacks and Queues

Note: Follow my naming exactly!

Part 1

Write a stack class and a queue class that each hold characters. Write a tree class that holds strings.

Bonus:
Use generics to allow the structures to hold any data type.

Part 2

Make a file / class called HomeworkFunctions

Level 1: hasCorrectDelimiters(String) -> Bool

Use a stack to write a function that verifies if a string has correctly paired brackets, parentheses and braces. These delimiters can be nested, so the most recent open delimiter needs to be closed before any of the previous delimiters can be closed.

The delimiters to check for are (), [], and {}

Ex:
The following are correct:

```
a = b + ( c - d ) * ( e - f )
a[ 1 ] = b[ c[ 2 ] ] + ( d + e ) * f
while ( a < ( b[ 3 ]  + c ) )
```

The following have mismatches:

```
a = b + ( c - d ) * ( e - f ) )
a[ 1 ] = b[ c[ 2 ] ] + ( d + e } * f
while ( a < ( b[ 3 ]  + c )
```

Level 2: isAPalindrome(String) -> Bool
Using this week's data structures, write a function that checks if a string is a palindrome. A palindrome is a string whose letters are the same forward and backward

Ex:
redivider
A Santa at Nasa
A dog! A panic in a pagoda!

Level 3: evaluateArithmaticExpression(String) -> Int?
Using this week's data structures, write a function that takes a string representing some integer arithmetic and evaluate it, returning nil if it was unable to parse correctly, or the answer if it was.

Ex:
"2 + 3" should return 5
"(2 + 3) * (4 + 6)" should return 50
"(2 + (3 * 4))" should return 14
"2 + 4 +" should return nil

Note: You can assume that all integers are 0-9 (although it would be fun to make it work for larger / negative numbers)
You can also ignore order of operations except for parens. For example you can have "2 * 3 + 4" return either 10 or 14, even though the latter is incorrect. But if the string is "(2 * 3) + 4" it should return 10, and if it's "2 * (3 + 4)" it should return 14. Obviously, the more accurate you can make this, the more challenging/rewarding!

Function signatures

All levels:

Swift

Stack	Queue
func push(element : Character)	func enqueue(element : Character)
func pop() -> Character?	func dequeue() -> Character?
func peek() -> Character?	func front() -> Character?
func clear()	func clear()
var size : Int	var size : Int
var isEmpty : Bool	var isEmpty : Bool

HomeworkFunctions
//Level 1
func hasCorrectDelimiters(_ : String) -> Bool
//Level 2
func isAPalindrome(_ : String) -> Bool
//Level 3
func evaluateArithmaticExpression(_ : String) -> Int?

C++

Stack	Queue
Stack()	Queue()
void push(const char element)	void enqueue(const char element)
char pop()	char dequeue()
const char peek() const	const char front() const
void clear()	void clear()
int size() const	int size() const
bool isEmpty() const	bool isEmpty() const

HomeworkFunctions
//Level 1
bool hasCorrectDelimiters(std::string string)
//Level 2
bool isAPalindrome(std::string string)
//Level 3
int evaluateArithmaticExpression(std::string)