Introduction to Data Structures
Assignment - 1
Due Date: 26th October 2013

Problem 1: Balanced Bracket Pairs

Given a set of different types of paired symbols; determine whether it is balanced or not (the opening and closing version of each type are paired correctly). Any other type of characters may appear in the input will be neglected.
For example:
(**(*[*])****) is balanced, also [{}**[]**()]** is balanced. But ((*{*)*}) is unbalanced as well as [()***[]***{*(*)}.
Write a C++ program that requires a string containing and expression as an input from the user and check for balanced bracket pairs using stack data structure.

Problem 2: Simple Addition Calculator

Given a number of digits your program is required to print the addition result of these numbers. Write a C++ program that simply add numbers (using stacks), the program reads in floating point numbers, push them onto a stack, add them together whenever a plus sign, "+", is entered, and print the result. In addition, the adding machine will recognize the following instructions:

- A comma "," means a "cancel the last entry".
- A period "." means "cancel all entries", that is clear the stack.
- Making sure that the input is either a floating number or "+" or "," or "." characters.

Problem 3: Palindrome

Using one or more stacks, write a C++ program to read in a string of characters and determine whether it forms a palindrome. A palindrome is a sequence of characters that reads the same both forward and backward. For example: aabbaa, abbcbba, and aaaaa are palindromes.
You must assume that the data are correct and the maximum numbers of input characters is 80.

Problem 4: Binary Convertor

Given a decimal number from the user your program should convert it to binary using stack's functions. Write a C++ program that asks the user to enter a decimal number and then prints to the user the binary version of the given number. i.e.: 22 (decimal) its binary is 10110.

Some guide tips for Problem 1

In our application, we consider parenthesis pairs: ( ), [ ], { }.
Any number of other characters may appear in the input, but a closing parenthesis symbol must match the last unmatched opening parenthesis symbol and all parenthesis symbols must be matched when input is finished.
Examples – Balanced expressions:

a. (XX(XX( ))XX)
b. [ ] ( ) { }
c. ({ } { XXX} XXX ( ) XXX)
d. ([{ [(([{X}]) X)] }X])

Examples – Unbalanced expressions:

a. (XX(XX( ))XXX)XXX)
b.] [
c. (XX[XXX)XX]

How the program works?

a. The program reads an expression character by character. For each character, it does one of three tasks depending on whether the character is:
    - An opening special symbol.
        - If the character is an opening special symbol, it is saved on the stack.
    - Or not a special symbol.
        - If the character is not a special symbol, it is discarded and another character is read.
    - A closing special symbol.
        - If the character is a closing special symbol, it must be checked against the last opening special symbol, which is on the top of the stack.
            - ❖ If they match, the character and the last opening special symbol are discarded and the program processes the next input symbol.
            - ❖ If the closing special symbol does not match the top of the stack or the stack is empty, then the expression is ill-formed.
b. When the program has processed all of the characters, the stack should be empty – otherwise extra opening special symbols are present.

Important Notes:

The assignment would be submitted through elearning ONLY

Deliverables:

1. Source Code your running code giving the required output (submit the .cpp file).
2. Document having a clear explanation for how you implemented your program and the steps you followed to get your code done (The code should be fully documented).