Assignment 3 Question 3

#include <iostream>

#include <stack>

#include <string>

using namespace std;

// Function to check if opening and closing brackets form a valid pair

bool ArePair(char opening, char closing) {

if (opening == '(' && closing == ')') return true;

else if (opening == '{' && closing == '}') return true;

else if (opening == '[' && closing == ']') return true;

return false;

}

// Function to check if all parentheses/brackets in the expression are balanced

bool AreParanthesesBalanced(string exp) {

stack<char> S; // stack to store opening brackets

// Traverse the expression character by character

for (int i = 0; i < exp.length(); i++) {

// If current character is an opening bracket, push it to stack

if (exp[i] == '(' || exp[i] == '{' || exp[i] == '[') {

S.push(exp[i]);

}

// If current character is a closing bracket

else if (exp[i] == ')' || exp[i] == '}' || exp[i] == ']') {

// Case 1: Stack is empty (no opening bracket to match with)

// Case 2: Top of stack doesn't match with current closing bracket

if (S.empty() || !ArePair(S.top(), exp[i])) {

return false; // Not balanced

} else {

S.pop(); // Valid pair found, pop the opening bracket

}

}

}

// Finally, if stack is empty → all brackets matched correctly

return S.empty() ? true : false;

}

int main() {

string expr = "{(]}"; // sample expression

// Call function and print result

if (AreParanthesesBalanced(expr))

cout << "Balanced\n";

else

cout << "Not Balanced\n";

return 0;

}