<u>Part 2: Lab Tasks (10 points)</u>

Note: Copy this section into a new word file then save it. You will only submit this section of the lab manual.

<u>Problem Description</u>

A bracket is considered to be any one of the following characters: (,), {, }, [, or].

Two brackets are considered to be a matched pair if the an opening bracket (i.e., (, [, or {) occurs to the left of a closing bracket (i.e.,),], or }) of the exact same type. There are three types of matched pairs of brackets: [], {},and ().

A matching pair of brackets is not balanced if the set of brackets it encloses are not matched. For example, {[(])} is not balanced because the contents in between { and } are not balanced. The pair of square brackets encloses a single, unbalanced opening bracket, (, and the pair of parentheses encloses a single, unbalanced closing square bracket,].

By this logic, we say a sequence of brackets is balanced if the following conditions are met:

- It contains no unmatched brackets.
- The subset of brackets enclosed within the confines of a matched pair of brackets is also a matched pair of brackets.

Given string of brackets, determine whether the sequence of
brackets is balanced. If a string is balanced, print ***YES***. Otherwise, print ***NO***.

Input Format

A single string ***s*** represents a sequence of brackets.

Output Format

Print ***YES*** or ***NO***.

Sample Input

{[()]}

Sample Output

YES

```cpp
#include <iostream>
#include <string>
#define MAX 30

using namespace std;

struct Stack
{
    char B[MAX];
    int top = -1;
};
struct array
{
    char B[MAX];
};

void push(Stack *s,char letter);
char pop(Stack *s);


int main(void)
{
    Stack a;
    array b;

    string check;
    cin >> check;

    if (check.size() %2 != 0)
        {
            cout << "no" << endl;
            return 0;
        }
    for (int i = 0; i < check.size() ; i++)
    {
        char item =check[i];
        push(&a , item);
        b.B[i] = item;
    }

    string here = "Yes";
    for (int i = 0 ; i < check.size()/2 ; i++)
    {
        char item = pop(&a);
        switch (item){
        case '}':
            if (b.B[i] != '{')
            here = "No";
            break;
        case ')':
            if (b.B[i] != '(')
            here = "no";
            break;
        case ']':
            if (b.B[i] != '[')
            here = "No";
            break;
        default:
            here = "No";
        }
    }
    cout << here << endl;
}
void push(Stack *s,char letter)
{
    if (s -> top == MAX)
    {
        cout << "overload" << endl;
    }
    else
    {
        s -> top++;
        s -> B[s -> top] = letter;
    }
}
char pop(Stack *s)
{
    if (s -> top == -1)
    {
        cout << "underload" << endl;
        return -1;
    }
    char item = s -> B[s -> top];
    s -> top--;
    return item;
}
```