Page No.
Date

## Q Redundant Brackets

Given valid mathematical expression in the form of a String. You are supposed to return true if the given expression contains a pair of redundant brackets else return false. The Given String only contains '(', ')', '+', '*', '-', '/' and lowercase letters.

Eg:- (a+b) ✓ Valid brackets → Redundant ✗

((a+b)) ✗ Invalid → Redundant ✓

Approach: if any operator between the brackets then return false else return true

Eg:- Dry Run

S = "(a+b*(c-d))"

(c-d) => (-) ✗

(a+b*()) ✗ => Redundant ✗

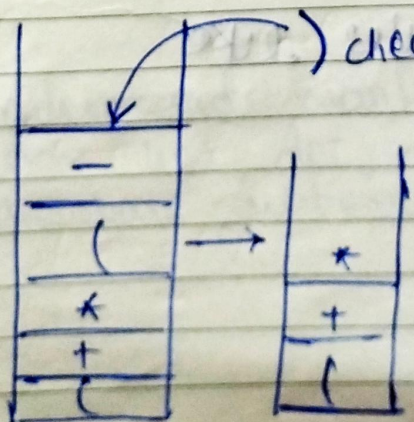
Code :-

```
#include <stack>
bool findRedundantBrackets(string &s)
{
    stack<char> st;
    for(int i=0; i<s.length(); i++){
        char ch = s[i];
        if( ch=='(' || ch=='+' || ch=='-' ||
            ch=='*' || ch=='/')
        {
            st.push(ch);
        }
        else {
        // ch is ')' or lowercase letter
            if( ch==')' ) {
                bool isRedundant = true;
                while(st.top() != '(' ) {
                    char top = st.top();
```

```
if ( top == '+' || top == '-' || top == '*' || top == '/')
{
    isRedundant = false;
}
    st.pop(); // To pop the operators
}

if ( isRedundant == true)
    return true;
st.pop(); // To pop the opening bracket
}
}
}
return false;
}
```

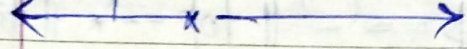
Dry Run : (a+b*((c-d))

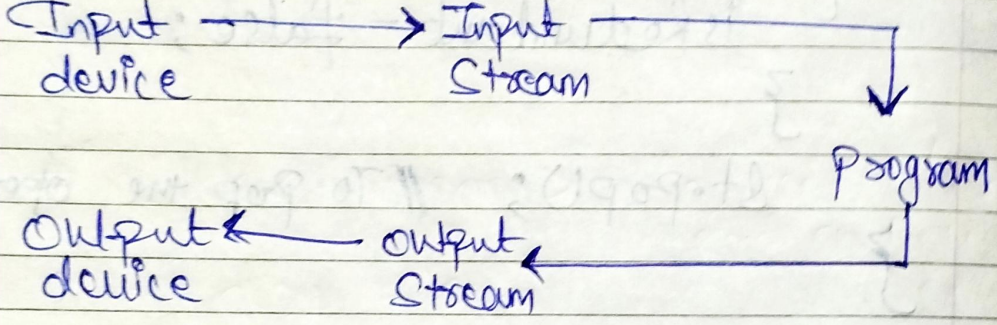


→ for more detail Explanation & understand visit c++ udemy / Stream section

Page No.
Date

## I/O streams






Common header files

iostream : provides definition for formatted input and output from/to streams

fstream : Provides definition for formatted input and output from/to file Stream

iomanip : provides definition for manipulator used to format stream I/O

... etc header files like

- ios
- ifstream
- ofstream
- fstream
- stringstream

Page No.
Date

## Global Stream objects

cin → Standard input stream

cout → Standard output stream

cerr → Standard error stream

clog → standard log stream

(*) formatting boolean type :-

We will be using the stream manipulator to print true or false rather than 1 and 0.

Eg:- cout << std::boolalpha;

cout << (10 == 10) << endl;

↳ true.

Q. understand manipulators for integer. like hex, dec, oct and so etc.
showbase, uppercase, lowercase ... etc.

Q. Read more about formatting floating point type

(*) #include<fstream>, is used for read and write for any file.

(*) <ifstream> & <ofstream> ?

(*) in_file & out_file ?

(*) Std::cerr ?

(*) #include<cstream> ?

Page No.
Date

## # Enumeration :-

An enumerated type provides a useful way to store a set of named integral constants known as enumerator

Eg:- Name of days in a week
{ Mon, Tue, wed ... }.

(A) It is uses to increase the Readability

Eg:-

```
Enum State { Engine_failure = 0, Inclement_
             weather = 1, Nominal = 2};

Enum Sequence { Abort = 3, Hold = 4, Launch};

int user_input;
std:: cin>> user_input;

State State = State(user_input);
if (State == Engine_failure)
    initiate (Abort);
else if (State == Inclement_weather)
    initiate (Hold);
else if (State == Nominal)
    initiate (Launch);
```

(*) C++ Enumeration Structure:-

enum-key enum-name : enumerator-type {}

enum-name → optional

enum-key → Define the scope of the enumeration

enumerator-type → can be omitted and the compiler will try to deduce it.

{} → list of the enumerators

Eg:-

```
enum { Red, Green, Blue };
       0    1     2
```

// Implicit Initialization will be done as the user doesn't assign the value explicitly.

Eg:- `enum { Red = 4, Green = 2, Blue = 7 };`

(*) Types of Enumeration