**ASSIGNMENT NO. 8**

**TITLE:** Checking parenthesis.

**PROBLEM STATEMENT:** In any language program mostly syntax errors

occurs due to unbalancing delimiters such as (),

{ },[ ].write a c++ program using stack to check

Whether given expression is well parenthesized

Or not.

**PRE-REQUISITES:** Basic operation of stack and knowledge about infix

Postfix and prefix expression and conversion rules.

**LEARNING OBJECTIVE:** Learning of stack operation using reversing

Of string and form balanced expression.

**THEORY:** Checking correctness of well formed parenthesis .consider Mathematical expression that includes several sets of nested Parenthesis .To ensure that parenthesis are nested correctly

We need to check:

1. Their are equal number of right and left parenthesis .

2. Every right parenthesis is preceded by matching left parenthesis.

In example :((x+y) or(x+y))

This expression is against condition 1.

And (x+y)-(or(x+y))(A-B)

This is against condition 2.

To solve this problem we need to define parenthesis count at a particular point .In expression as number of left parenthesis equal to number of right parenthesis.

It can be possible in left to right scanning expression.

2 conditions that must hold parenthesis in expression

1. Parenthesis count at each point is non- negative .

2. Parenthesis count at end of the expression is zero.

CHECKING PROCEDURE:

* Checking process declare a character scan
* Now travels through expression or string

a. If current character is (‘(‘, ’{‘ ,’[‘) then push it to stack.

b. If the current character is(‘)’ ,’}’ ,’]’) then pop from stack & match the element parenthesis.

If match found then pop from the stack after travels if there is same starting bracket left in stack then given expression is not balanced .

**Analysis:**

* There are five types of input character

1. Opening brackets

2 .Numbers

3. Operators

4. Closing bracket

5. Newline character

**ALGORITHM:**

1. Read a character input.

2. Actions to be performed at the end of each input

i) Opening bracket then push into stack and go to step 1.

ii) If it I number print it and go to step 1.

iV) closing bracket then pop it from stack.

v) If it is operator print it and go to step 1

vi)If the pop element is opening bracket discard it & go to step 1.