Practical No -10

Title: Stack Concept

Aim: In any language program mostly syntax error occurs due to unbalancing delimiter such as (),{},[]. Write C++ program using stack to check whether given expression is well parenthesized or not.

Prerequisite:

- Basics understanding of stack

Objectives:

- To understand implementation of Stack

Input: Any expression

Outcome:

- At end of this experiment, student will be able to illustrate the concept to design all the aspects of stack & its variants

Theory:

Stacks

Stack is a LIFO (Last In First Out) data structure. It is an ordered list of same type of elements. A stack is a linear data structure where all insertions and deletions are permitted only at one end of the list. When elements are added to stack it grows at one end. Similarly, when elements are deleted from a stack, it shrinks at the same end.

Stack As an ADT

Stack is a LIFO structure. Stack can be represented using an array. A one dimensional array can be used to hold elements of a stack. Another variable "top" is used to keep track on the index of the top element.

Formally, a stack may be defined as follows:

```
typedef struct stack

{

 int data[MAX];

 int top;
```

};

Basic operations on Stack:

1. **void initialize (stack *P)** : It initializes a stack as an empty stack. Initial value of stack is set to -1.

```
void initialize (stack *P)
{
P -> top = -1;
}
```

2. **int empty (stack *P)**: Function checks whether the stack is empty. It returns 1 or 0 depending on whether the stack is empty or not.

```
int empty ( stack *P)
{
        if (P -> top ==-1)
        return (1);
        return (0);
        }
```

3. **int full (stack *P)**: Function checks whether the stack is full. Whenever the stack is full, top points to the last element (ie. MAX-1) of the array. It returns 1 or 0 depending on whether the stack is full or not.

```
int empty ( stack *P)
{
if (P -> top == -1)
  return (1);
  return (0);
}
```

4. int push (stack *P, int x):

The function inserts the element x onto the stack pointed by P. Insertion will cause an overflow if the stack is full.

```
void push ( stack *P, int x)
{
   P -> top = P -> top + 1;
  P -> data[ P -> top] = x;
}
```

5. int pop (stack *P): The function deletes topmost element from the stack and also returns it to the calling program. Deletion from an empty stack will cause underflow.

```
int pop ( stack *P)
{
int x;
x = P -> data[ P -> top];
P -> top = P -> top -1;
return (x);
}
```

Algorithm:

Declare a character stack S.
Now traverse the expression string exp.

-If the current character is a starting bracket (**'(' or '{' or '['**) then push it to stack.
-If the current character is a closing bracket (**')' or '}' or ']'**) then pop from stack and if the popped character is the matching starting bracket then finish else brackets are not balanced.

After complete traversal, if there is some starting bracket left in stack then "not balanced

Conclusion:

Thus we have implemented C++ program for to check given expression is well parenthesized or not using stack