Date-

Assignment No. :

Problem Statement:

Program in C to check whether a string is balanced or not from algebraic sense.

Theory:

While doing calculations in Algebra, there are certain expressions which are enclosed with various types of brackets to denote which operations are to be done in which order. This program will check wether the brackets are given in right order or not, have right closing or not. The brackets are considered as a string in our program.

For example, the program should print “Balanced” for expression = “[()]{}{[()()]()}” and “string not Balanced” for exp = “[(])”.

Algorithm:

Input Specification: An array of characters which represents the expression, say expr.

Output Specifictaion: The given expression is balanced in algebraic sense or suitable unsuccessful message.

Data Structure Used: Two arrays expr and stack, used for storing the string given as input by the user and as a temporary stack, respectively.

Steps:

Algorithm for method Opening(x):

Step 1: If(x == "]")
    a. Return "["
Step 2: Else If(x == "}")
    a. Return "{"
Step 3: Else If(x == ")")
    a. Return "("
    b. [End of if structure]

Algorithm for method Closing(x):

Step 1: If(x == "[")
    a. Return "]"
Step 2: Else If(x == "{")
    a. Return "}"
Step 3: Else If(x == "(")
    a. Return ")"
    b. [End of if structure]

Algorithm for method Main():

Step 1: Print "Enter the expresion : "
Step 2: Input expr
Step 3: Set i = 1, sp = 1
Step 4: Repeat through step 4.a to 4.c while (expr[i] != Null)
    a. If(expr[i] == "(" Or expr[i] == "{" Or expr[i] == "[")
    b. Then
        i. Set stack[sp] = expr[i]
        ii. Set sp = sp + 1
    c. Else If(expr[i] == ")" Or expr[i] == "}" Or expr[i] == "]")
    d. Then
        i. If(sp == 1)
        ii. Then

1. Print "\nUnexpected closing brace : '", expr[i], "'"
2. Set sp = 2
3. Break

iii. Else If(stack[sp - 1] != Opening(expr[i]))
iv. Then

1. Print "\nExpected closing : '", Closing(stack[sp - 1]), "'"
2. Print "\nReceived : '", expr[i], "'"
3. Break

v. Else
1. Set sp = sp - 1

vi. [End of inner if structure]

e. [End of outer if structure]

f. Set i = i + 1
g. [End of while loop]

Step 5: If(sp == 1)
a. Print "\nGiven expression is balanced in terms of paranthesis!"
Step 6: Else
a. Print "\nGiven expression is not balanced in terms of paranthesis!"
b. [End of if structure]

Source Code:

```
#include <stdio.h>
#include <string.h>
#include <stdint.h>

char closing(char open){
   return open == '(' ? ')' : open == '[' ? ']' : '}';
}

char opening(char closing){
   return closing == ')' ? '(' : closing == '}' ? '{' : '[';
```

```
}

int main(){
   char expr[100];
   printf("\nEnter the expression : ");
   fgets(expr, 98, stdin);
   size_t len = strlen(expr), i = 0;
   char stack[len];
   uint64_t sp = 0;
   while(i < len){
      switch(expr[i]){
         case '(':
         case '{':
         case '[':
            stack[sp++] = expr[i];
            break;
         case ')':
         case '}':
         case ']':
            if(sp == 0){
               printf("\nUnexpected closing brace : '%c'\n", expr[i]);
               sp = 1;
                 break;
            }
            if(stack[sp - 1] != opening(expr[i])){
               break;
            }
            else
               sp--;
      }
      i++;
   }
   if(sp == 0)
```

```
    printf("\nThe given expression is valid in terms of paranthesis!\n");
  else
    printf("\nThe given expression is not valid in terms of paranthesis!\n");
  return 0;
}
```

Input & Output:

Set 1:

```
Enter the expression : 12+14(90-15)
The given expression is valid in terms of paranthesis!
--------------------------------
```

Set 2:

```
Enter the expression : {56-(65*98+78)[54-90])
The given expression is not valid in terms of paranthesis!
```

Discussion:

1. This program uses array to store the strings and then check its validity. For larger strings there is a possible buffer overflow.
2. The complexity of this program is high as it is upon the user to give the string as input. So if the user gives a large string and in our program we haven't declared a large array, then it will be problematic. Although if we declare a large array, but the user gives a short string as input, then there will be problem of memory wastage.