LAB Exp No: 2

Problem Statement:

Implement an effective solution for Balanced parenthesis problem.

.

Aim:

To write the algorithm and program for the following problem using Stack data type..

Algorithm:

```
STEP 1 : START
STEP 2: Make a character stack declaration.
STEP 3: Now look through the exp. expression string.
STEP 4: If the current character is a beginning bracket('(',", or'['), stack it.
STEP 5:If the current character is a closing bracket(') or'' or']', pop from the
stack; if the popped character is the matching starting bracket, great; otherwise,
the brackets are unbalanced.
STEP 6: If there are any starting brackets left in the stack after traversal, the stack
is said to be "unbalanced."
STEP 7: END
```

Program:

```
#include <bits/stdc++.h>
using namespace std;

bool areBracketsBalanced(string expr)
{
  stack<char> s;
  char x;

  for (int i = 0; i < expr.length(); i++)
  {
    if (expr[i] == '(' || expr[i] == '['
      || expr[i] == '{')
    {
      s.push(expr[i]);
      continue;
    }
```

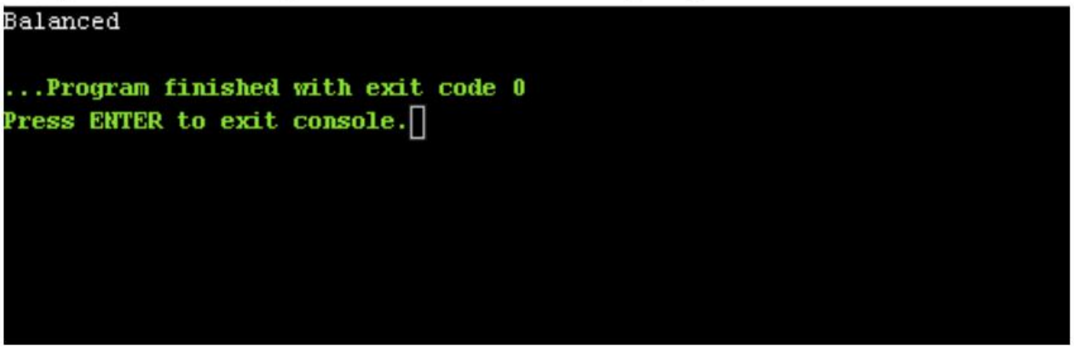
```
        if (s.empty())
            return false;

        switch (expr[i]) {
        case ')':
            x = s.top();
            s.pop();
            if (x == '{' || x == '[')
                return false;
            break;

        case '}':
            x = s.top();
            s.pop();
            if (x == '(' || x == '[')
                return false;
            break;

        case ']':
            x = s.top();
            s.pop();
            if (x == '(' || x == '{')
                return false;
            break;
        }
    }
    return (s.empty());
}
int main()
{
    string expr = "{()}[]";
    if (areBracketsBalanced(expr))
        cout << "Balanced";
    else
        cout << "Not Balanced";
    return 0;
}
```

Output: The above problem has the following Ouput.



```
Balanced

...Program finished with exit code 0
Press ENTER to exit console.
```

Exp. No: 2

Problem Statement:

For reversing the string by using stack implementation.

Aim:

To reversing the string by using stack implementation.

Algorithm:

Step 1: define and array of character and a variable top.

Step 2: then declara three functions push pop and display.

Step 3: Then ask the user for inserting the values.

Step 4: Then check the necessary condition for each element to be used in functions.

Step 5: And then display the result as a reversed string

Program: (c++)

```
#include <iostream>
using namespace std;
char stack[100], n=100, top=-1;
void push(int val) {
  if(top>=n-1)
  cout<<"Stack Overflow"<<endl;
  else {
    top++;
    stack[top]=val;
  }
}
void pop() {
  if(top<=-1)
  cout<<"Stack Underflow"<<endl;
  else {
    cout<<"The popped element is "<< stack[top] <<endl;
    top--;
  }
}
void display() {
  if(top>=0) {
    cout<<"the reversed string is";
    for(int i=top; i>=0; i–)
    cout<<stack[i]<<" ";
    cout<<endl;
  } else
  cout<<"Stack is empty";
}
int main() {
  int ch;
  char val;
  cout<<"1) Push in stack"<<endl;
  cout<<"2) Pop from stack"<<endl;
  cout<<"3) Display the reversed string"<<endl;
  cout<<"4) Exit"<<endl;
  do {
    cout<<"Enter choice: "<<endl;
    cin>>ch;
    switch(ch) {
      case 1: {
        cout<<"Enter value to be pushed:"<<endl;
        cin>>val;
        push(val);
        break;
      }
      case 2: {
        pop();
        break;
      }
      case 3: {
        display();
        break;
      }
      case 4: {
        cout<<"Exit"<<endl;
        break;
      }
      default: {
        cout<<"Invalid Choice"<<endl;
      }
    }
  }while(ch!=4);
  return 0;
}
```

Output:

```
1) Push in stack
```

```
2) Pop from stack
3) Display the reversed string
4) Exit
Enter choice:
1
Enter value to be pushed:
d
Enter choice:
1
Enter value to be pushed:
e
Enter choice:
1
Enter value to be pushed:
v
Enter choice:
1
Enter value to be pushed:
e
Enter choice:
1
Enter value to be pushed:
s
Enter choice:
1
Enter value to be pushed:
h
Enter choice:
1
Enter value to be pushed:
c
Enter choice:
2
The popped element is c
Enter choice:
3
the reversed string ish s e v e d
Enter choice:
4
Exit
```

Result:

the problem given above has been solved using stack implementation