Military Institute of Science and Technology
Department of Computer Science and Engineering
Level-2, Spring Term
CSE-204 (Data Structures and Algorithms 1)
Date: 06 May 2021

Time: 25 mins **Full Marks: 20**

Statement:
Rohit is really serious about programming. He does it all day long. He uses the C++ language for programming. But sometimes he faces problems with curly braces while programming. He often gets this error that he didn't close the statements properly.
So, he starts thinking about solving this problem. He remembers that last week in the Data Structures and Algorithms Sessional, Lec Shahriar taught about Stack. Suddenly a solution came to his mind, what if he uses Stack to check for balanced parenthesis. He starts looking for an algorithm for it. Finally, he found one which is given below:

Algorithm

- Traverse the input expression string from left to right.
    1. If the current character of the expression is a starting bracket (**'(' or '{' or '['**) then push it into a Stack ***S***.
    2. If the current character of the expression is a closing bracket (**')' or '}' or ']'**) then perform a pop operation on the Stack ***S***. If the popped character is the corresponding starting bracket of the current character then continue the process otherwise finish the process with a message "Not Balanced".
- After the complete traversal, if the stack is empty then finish the process with a message "Balanced", otherwise finish the process with a message "Not Balanced".

Basic Instructions:

1. You have to solve this problem using the implemented stack given in your assignment, which means you have to use the push and pop functions of the implemented stack.
2. There will be a little bit of change in your stack class as you have to work with characters here, not integers. So keep that in mind.
3. You have to implement a boolean function ***bool areBracketsBalanced(string exp),*** where exp represents the string expression and taken as input from the user.
4. Create a separate menu with ***choice = 6*** for executing the ***areBracketsBalanced*** function.

Sample:

Example No	Input	Output
Example-1	( )	Balanced
Example-2	{( )( )}	Balanced
Example-3	{( )}[ ]	Balanced
Example-4	[ ] ( )]	Not Balanced
Example-5	[{}( ]( )	Not Balanced
Example-6	)(	Not Balanced