1. **Stack Implementation:**
   * The code defines a stack with a fixed size using an array **stack** and a variable **top** to keep track of the top element in the stack.
   * Functions **stackFull** and **stackEmpty** check whether the stack is full or empty, respectively.
   * Functions **push** and **pop** perform the standard push and pop operations on the stack.
2. **Delimiter Analyzer Function (analyser):**
   * The function takes three parameters: a string **s1** representing the code snippet, and two strings **open** and **close** representing the opening and closing delimiters.
   * It uses a stack to check if the delimiters in the code snippet are balanced.
   * It iterates through each character in the code snippet, and when an opening delimiter is encountered, it is pushed onto the stack.
   * When a closing delimiter is encountered, it checks whether the corresponding opening delimiter is at the top of the stack. If not, it indicates a syntax error.
   * The function returns 1 if there is a syntax error or 0 if the delimiters are balanced.
3. **main Function:**
   * Initializes a string **s1** with a code snippet.
   * Defines strings **open** and **close** representing opening and closing delimiters.
   * Calls the **analyser** function to check whether the delimiters are balanced or not.
   * Displays the original code and prints whether the delimiters are balanced or if there is a syntax error.
4. **Example Code:**
   * The example code snippet in **s1** is a C++ code with nested if-else statements and arithmetic expressions.
5. **Testing with Different Delimiters:**
   * The code is tested with both **{**, **}**, **(**, **)**, **[**, **]** as delimiters, and **{{**, **}}**, **((**, **))**, **[[**, **]]** as an example of mismatched delimiters.
6. **Note:**
   * The code checks for balanced delimiters by using a stack, ensuring that each opening delimiter has a corresponding closing delimiter.

Algorithm:

1. \*Include necessary libraries and define constant SIZE for the stack.\*

2. \*\*Declare a global integer array stack to simulate a stack and initialize top to -1.\*\*

3. \*\*Define functions to check if the stack is full (stackFull) or empty (stackEmpty).\*\*

4. \*\*Define functions push and pop for stack operations.\*\*

5. \*\*Define a function analyser to check the balance of delimiters in the given code snippet.\*\*

6. \*\*Define the main function to create a sample code snippet and call the analyser function.\*\*