Code Trace Process

Purpose	- A code trace is a method for hand simulating the execution of your code in order to manually verify that it works correctly before test.
Resources	- Source code to be verified - A test case: input and expected output values. - Blank Code Trace table (next page)
Entry Criteria	- Source code compiles successfully
General notes	- Currently there are no quantitative process or quality measures.

Step	**Activities**	**Description**
1	Line numbering	- In the left margin of the source code printout, sequentially number each executable statement in the source code.
2	Initialize Table	- Place the name of each variable in the program in a separate column under the "Variables" header in the table. - Write each boolean condition in the program in a separate column under the "Conditions" header in the table. - Write the input value from the test data in the first row under the appropriate column.
3	Trace	- Starting with statement 1, simulate the action the computer will take when it executes each statement. - Write the statement number in the "Stmt" column. - For assignment statements, write the value that is assigned to the variable in the corresponding column. - For decisions, evaluate the boolean condition and write T or F in the corresponding column. - Repeat until the program terminates, or it becomes clear it is stuck in an infinite loop.
4	Verify	- Compare the return value (or other output) with the expected output from the test data. - If they match, write "Verified" at the bottom of the table. - If there is a discrepancy, write "Not verified" at the bottom of the table and circle the output value that the program produced.

Exit Criteria	- The simulation has either completed or is stuck in an infinite loop. - Each variable column has the correct values filled in. - Rows are numbered with the correct statement executed. - The result is noted at the bottom of the table.
Output	- A numbered source code listing. - A completed trace table. - The result of verification.

	Variables					Conditions	
Stmt							

EXAMPLE

The table below illustrates how the following code would be traced for this test data:

Input: `extract(“ABCD”)` **Expected Output:** `“BC”`

```
       public String extract(String text)
       {
          int position;
          int length;
          // Is length odd or even?
1         length = text.size();
2         if (length % 2 == 1)
          {
3             position = length / 2;
4             length = 1;
          }
          else
          {
5             position = length / 2 - 1;
6             length = 2;
          }
7         String result = text.substring(position, position + length);
8         return result;
       }
```

	Variables					Conditions
Stmt	text	position	length	result		`length % 2 == 1`
	“ABCD”					
1			4			
2						F
5		1				
6			2			
7				“BC”		

EXERCISE

Below is the source code for the GCD class which has been compiled successfully.
Follow the process script for the Code Tracing Process to verify that the code works as intended. (If there are details about the process which seem unclear to you, write down any assumptions you make).

Test Data
findGCD(10,25)   Expected Output: 5

```
/**
 *  Greatest Common Divisor utility class
 */
public class GCD
{
   /** Find the GCD of two integers.  */
   public static int findGCD(int num1, int num2)
   {
      while (num1 != num2)
      {
         if (num1 > num2)
         {
            num1 = num1 - num2;
         }
         else
         {
            num2 = num2 - num1;
         }
      }
      return num1;
   }
}
```

	Variables					Conditions	
Stmt							