Write an algorithm in psuedocode which reads five numbers from the user, and displays the average of those numbers.

What testcases might you use to verify that your program works?

**inputs**: 5 numbers  
**outputs**: result  
**errors**: none

* **SET**count to zero
* **SET**sum to zero
* **WHILE**count < 5
  + **READ**number as A
  + **COMPUTE**sum **AS sum** + A
* **COMPUTE**result **AS**sum / 5
* **DISPLAY**result

Testcases:  
[input][output]  
[0,0,0,0,0][0]  
[1,2,3,4,5][3]  
[-1,1,-1,1,-1][-0.2]

Pretend there is a combination lock which has three dials, each of which can go from 0 to 9. The combination to unlock this lock is 6-1-4.

Write code which will read three numbers from the user, and prints "unlocked" if their three numbers are the combination, and "locked" otherwise. Use the IO class for input and output.

What testcases might you use to test if your algorithm works?

**inputs**: 3 numbers  
**outputs**: "unlocked" or "locked"  
**errors**: none

public class Lock  
{  
  public static void main( String [] args )  
  {  
    System.out.println( "Enter the first number: " );  
    int firstNum = IO.readInt();  
    System.out.println( "Enter the second number: " );  
    int secondNum = IO.readInt();  
    System.out.println( "Enter the third number: " );  
    int thirdNum = IO.readInt();  
  
    if( firstNum == 6 && secondNum == 1 && thirdNum == 4 )  
      IO.outputStringAnswer( "unlocked" );  
    else  
      IO.outputStringAnswer( "locked");  
  }  
}

[input][output]  
[6,1,4]["unlocked"]  
[4,1,6]["locked"]  
[1,2,3]["locked"]  
[614,614,614]["locked"]  
[-6,-1,-4]["locked"]

The factors of a number, n, are all of the positive numbers that n is evenly divisible by, including itself.

As an example, the factors of 18 are 1, 2, 3, 6, 9, and 18.

Write an algorithm in psuedocode that prints all of the factors of a number. You may assume that all numbers will be positive.

What testcases might you try to verify that your program works?

* **READ**the number **AS**number
* **SET**current **TO**1
* **WHILE**current is less than or equal to number **DO**
  + **SET**remainder **TO**number modulus current
  + **IF**remainder is equal to 0 **THEN**
    - **PRINT**current
  + **ENDIF**
  + **INCREMENT**current
* **ENDWHILE**

[input][output]  
[18][1,2,3,6,9,18]  
[1][1]  
...

For each of the following, indicate whether there will be a run-time error, compile-time error, or no error. 

1. int x = 6 / 0;
2. System.out.println(8 / 0);
3. System.out.println(8.0 / 0.0);
4. System.Out.println("Hello world!");
5. System.out.println("hello world);
6. int j = 13.5;
7. char c = 'l';
8. double r = (int) 6.0;
9. int q = 6 + 10
10. int a = 5;  
    int b = 5;  
    int result = 1 / ( a - b );

1. **Run time error**: Division by zero

2. **Run time error**: Division by zero

3. **No error**: Weird, but floating point numbers allow this. The result is infinity :(

4. **Compile time error**: This is a syntax error, the O in Out is not meant to be capitalized.

5. **Compile time error**: This is a syntax error, there is no closing quotation marks

6. **Run time error**: Loss of precision, putting a double into an int without a cast.

7. **No error**

8. **No error**

9. **Compile time error**: Missing semicolon, syntax error

10. **Run time error**: Division by zero

The following code snippet is meant to find the sum of all the numbers between 1 and 100. Find three **logical** errors:

int sum = 0;

for(int i = -1; i > 100; i ++){

sum -= i;

}

int sum = 0;

for(int i = 1; i < 100; i ++){

sum += i;

}

What is a method signature? What are its elements?

Write the method signature for a method which takes in 2 doubles and returns their average.

Write the method signature for a method which takes in the user's name and age and returns true if they are an adult.

Write the method signature for a method which takes in the user's first name and last name and returns their name as "last, first".

* A method signature is akin to a recipie listing on a website: it may tell you that if you input 2 eggs, flour, sugar, and milk, you will get a cake, but it *does not* tell you how to make that cake. It simply explains that if you give inputs of specific types, you will get an output of a specific type.  
    
  Method signatures let the caller of a method know what arguments to give to the method and what type they will get in return (if any). Method signatures take the form of:  
    
  public static (*parameter\_type parameter\_name*);
* public static double avg(double num1, double num2);
* public static boolean isAdult(String name, int age);
* public static String formatName(String first, String last);

A perfect number is defined as a number which is equal to the sum of all of its factors (excluding itself). As an example:

The factors of 28 are: 1, 2, 4, 7, 14, 28. Excluding 28, the sum of its factors is: 1 + 2 + 4 + 7 + 14 = 28.

Write a program to find all perfect numbers between 1 and 10,000.

for(int i = 1; i < 10000; i ++){

// Create a running sum of all of i's factors.

int sum = 0;

for(int j = 1; j < i; j ++){

if(i % j == 0) sum += j;

// Add j to sum if it is divisible into i.

}

if(sum == i) System.out.println("Perfect Number: " + i);

}

Write a program that reads in a number, *n*, via the **IO**module, and then prints a *vertical* pyramid to the console of height *n*:

Ex: height = 5:

    \*  
   \*\*\*  
  \*\*\*\*\*  
 \*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*

int n = IO.readInt();

for(int i = 1; i <= n; i++){

for(int j = 0; j < n-i; j++){

System.out.print(" ");

}

for(int j = 0; j < 2\*i-1; j++){

System.out.print("\*");

}

System.out.println();

}

For the following code snippet:

1. What is the output of the code snippet?
2. How many total times does the loop run?
3. In words, what does the snippet do?

int[] array = {4, 5, 6, 7, 8};

int mystery = 0;

for(int i = 0; i < array.length; i ++){

mystery += array[i];

}

int mystery2 = array.length;

System.out.println((mystery \* 1.0) / mystery2);

1. 6
2. 5, as the input array has 5 elements.
3. This code snippet computes the average of the array: first, it sums every element in the array, and then prints that sum divided by the number of elements.

BONUS: Arrays

Write a program which takes an array of integers as a parameter, and returns the average via IO.

public static void main(String[] args){

int[] arr; // This needs to be initialized somehow double sum = 0;

for(int i = 0; i < arr.length; i++){

sum += arr[i];

}

IO.outputDoubleAnswer(sum/arr.length);

}

Write a method, mode, which takes an integer array as parameter, and returns the single mode of the array. If there is no mode, return -1. You may assume all integers in the input array are positive.