You cannot use multiple return statements in any method submitted for this project. Additionally, the use of var is not permitted. **Each lab must use a loop to accomplish the assigned task**. No credit will be given for using existing methods (thereby avoiding the need to write the programming statements) to accomplish the assigned tasks.

Test 1 – How many multiples

```
public static int Test1(int start, int end, int factor)
```

Given three int's, start, end and factor, using a loop, count how multiples of factor occur between start and end (inclusive). A multiple is a number that is the factor multiplied by a whole number.

Example input

```
11, 28, 4
```

Example output

```
5
```

Test 2 – Factorial

```
public static int Test2(int number)
```

Given an int, `number`, calculate the factorial for `number`. A factorial is the result of multiplying all integers between 1 and the target (`number`) (so 3 factorial, 3!, is 3 X 2 X 1 = 6)

Example input

```
4
```

Example output

```
24
```

Test 3 – Raise to a power

```
public static int Test3(int root, int exponent)
```

Given two int's, `root` and `exponent`, use a loop to raise `root` to `exponent`. You are not allowed to use Math.Pow to compute the answer – you must use a loop. Additionally, `exponent` will be greater than or equal to 0 (no negative exponents)

Example input

```
6, 3
```

Example output

```
216
```

Test 4 – Consumable

```
public static int Test4(double onHand, double consume)
```

Given two doubles (`onHand` and `consume`) that represent the amount of an on-hand resource (`onHand`) and the amount of the resource that is `consume`d per cycles (think a time period). You are to determine how many cycles (time periods), before the on-hand amount is used up. For example, if you have 1 gallon of milk (`onHand`)and use 1 quart (1/4 gallon) per day (`consume`), determine when you expect to run out of milk (4 days). Return the number representing when the on-hand amount reaches 0 (or becomes negative). You may find it necessary to use the Round method from the Math class to avoid floating-point errors. I'd suggest rounding the results of each computation to 3 decimal places.

Example input

```
47.7, 0.9
```

Example output

```
53
```

Test 5 – Prime number

```
public static bool Test5(int number)
```

Given an int, `number`, determine if `number` is a prime number. Recall prime numbers are numbers that are divisible on by 1 and themselves. By definition negative values, 0 and 1 are not prime numbers (2 is the smallest prime number). If `number` is prime, return `true`, otherwise return `false`. You must check all values.

Example input

```
41
```

Example output

```
true
```

Test 6 – Build a string

```
public static string Test6(char starter, int number)
```

Given a char, `starter`, and an int, `number`, create a string the is made up of `starter` the `number` of characters (in order). For example, if `starter` is 'A' and `number` is 5, the string should be ABCDE (the starting value of A followed by the next 4 letters, in order)

Example input

```
g, 3
```

Example output

```
ghi
```

Test 7 – Making change

```
public static int Test7(double amount)
```

Given a double, `amount`, that represents an `amount` of US money, determine the minimum number of coins required to equal that `amount`. Use on quarters, dimes, nickels and pennies. There are two approaches to this problem. The first uses division and the second uses loops. You are not allowed to solve using division. For example, given 2.63, you would need 10 quarters, 1 dime and 3 pennies, for a total of 14 coins. You will probably find it necessary to use the Round method from the Math class to avoid floating-point errors. Since you are dealing with US currency/coins, it is suggested to round the results of each computation to 2 decimal places.

Example input

```
0.41
```

Example output

```
4
```

Test 8 – Return the multiples

```
public static string Test8(int factor, int qty)
```

Given two int's `factor` and `qty`, find the first `qty` multiples of `factor` (including factor as the first multiple) and add each to a string (each multiple will be followed by a single space to separate the multiples). For example, given 5 (`factor`) and 7 (`qty`), the resulting string would be 5 10 15 20 25 30 35

Example input

```
7, 4
```

Example output

```
7 14 21 28
```

Test 9 – Sum of values

```
public static int Test9(int start, int end)
```

Given two int's `start` and `end`, find the sum of all values between `start` and `end` (inclusive). For example, given 11 (`start`) and 19 (`end`), the result is 135

Example input

```
5, 9
```

Example output

```
35
```

Test 10 – Sum of values

```
public static int Test9(Random gener, int min, int max, int qty)
```

Given an instance of the Random class, `gener`, use it to generate `qty` random integers in the range `min` (inclusive) to `max` (exclusive). Find the and return the sum of the random numbers you generate. For example, given gener, 3, 7, 5, you will generate 5 random numbers between 3 and 7 (actually 3 and 6 because 7 is excluded) and find the sum of those 5 numbers. Return the sum

Example input

```
gener, 3, 7, 5
```

Example output

```
23
```