1. Given two integers, *a* and *b,* find the inclusive sum of all the integers between them and return it. If *a* equals *b*, return either *a* or *b*. Note that *a* and *b* are not ordered and that integers can be either positive or negative.

Example:

GetSum(1, 0) == 1   // 1 + 0 = 1

GetSum(1, 2) == 3   // 1 + 2 = 3

GetSum(0, 1) == 1   // 0 + 1 = 1

GetSum(1, 1) == 1   // 1 Since both are same

GetSum(-1, 0) == -1 // -1 + 0 = -1

GetSum(-1, 2) == 2  // -1 + 0 + 1 + 2 = 2

public int GetSum(int a, int b)

{

//write your code here!

}

1. Your function receives an array of positive integers as input. Your task is to determine whether the numbers are in **ascending** order.

public static bool IsAscOrder(int[] arr)

{

//write your code here!

}

1. You are given an array of integers. Your job is to take that array and find an index, *n*, where the sum of the integers to the left of *n* is equal to the sum of the integers to the right of *n*. If there is no such index that would make this happen, return -1.

Let's say you are given the array {1,2,3,4,3,2,1}:  
Your function will return the index 3, because at the 3rd position of the array, the sum of left side of the index ({1,2,3}) and the sum of the right side of the index ({3,2,1}) both equal 6.

Let's look at another one.  
You are given the array {1,100,50,-51,1,1}:  
Your function will return the index 1, because at the 1st position of the array, the sum of left side of the index ({1}) and the sum of the right side of the index ({50,-51,1,1}) both equal 1.

public static int FindEvenIndex(int[] arr)

{

//write code here!

}

1. You have a sum of money, *p*, that you want to invest. You want to know how many years, *y*, this sum has to be kept in the bank in order for this sum of money to amount to a predetermined dollar amount, *d*. The sum is kept for *y* years in the bank where interest, *i*, is paid yearly, and the new sum is re-invested yearly after paying tax, *t*. Note that the principal is not taxed; only the year's accrued interest is taxed.

Example:

Let P be the Principal = 1000.00

Let I be the Interest Rate = 0.05

Let T be the Tax Rate = 0.18

Let D be the Desired Sum = 1100.00

After 1st Year: P = 1041.00

After 2nd Year: P = 1083.86

After 3rd Year: P = 1128.30

Thus you have to wait 3 years for the initial principal to amount to the desired sum.

Your task is to complete the method provided and return the number of years, *y*, as a whole in order for you to get your desired sum.

Assumptions: Assume that Desired Principal 'D' is always greater than the initial principal, however it is best to take into consideration that if the Desired Principal 'D' is equal to Principal 'P' this should return 0 Years.

public static int CalculateYears(Double principal, Double interest, Double tax, Double desiredPrincipal)

{

//write your code here!

}