1. An array is said to be dual if it has an even number of elements and each pair of consecutive even and odd elements sum to the same value. Write a function named isDual that accepts an array of integers and returns 1 if the array is dual, otherwise it returns 0.

If you are programming in Java or C#, the function signature is

int isDual(int[ ] a)

If you are programming in C or C++, the function signature is

int isDual(int a[ ], int len) where len is the number of elements in the array

Examples

if the input array is return

{1, 2, 3, 0} 1 (because 1+2 == 3+0 == 3)

{1, 2, 2, 1, 3, 0} 1 (because 1+2 == 2+1 == 3+0 == 3)

{1, 1, 2, 2} 0 (because 1+1 == 2 != 2+2)

{1, 2, 1} 0 (because array does not have an even number of elements)

{} 1

private int IsDual(int[] a)

{

if (a.Length % 2 != 0 || a.Length == 2)

return 0;

else if (a.Length == 0)

return 1;

else

{

for (int i = 2; i < a.Length; i = i+2)

{

if (a[0]+ a[1] != a[i] + a[i + 1])

{

return 0;

}

}

return 1;

}

}

2,Given a positive integer k, another positive integer n is said to have k-small factors if n can be written as a product u\*v where u and v are both less than k. For instance, 20 has 10-small factors since both 4 and 5 are less than 10 and 4\*5 = 20. (For the same reason, it is also true to say that 20 has 6-small factors, 7-small factors, 8-small factors, etc). However, 22 does not have 10-small factors since the only way to factor 22 is as 22 = 2 \* 11, and 11 > 10.

Write a function has with signatuare boolean hasKSmallFactors(int k, int n) which returns true if n has k-small factors. The function should return false if either k or n is not positive.

Examples:

Input Output

10, 20 true (since 5\*4 = 20 and 5<10, 4< 10)

10, 22 false (since 22 can only be factored as 2\*11 and 11>10)

0, 5 false (since 0 is not positive)

private bool hasKSmallFactors(int k, int n)

{

if (k <= 0 || n <= 0)

return false;

else

{

for (int i = 1; i < k; i++)

if (n % i == 0)

if (i < k && n / i < k)

return true;

}

return false;

}

3. Define an array to be a Martian array if the number of 1s is greater than the number of 2s and no two adjacent elements are equal. Write a function named isMartian that returns 1 if its array argument is a Martian array, otherwise it returns 0.

There are two additional requirements.

You should return 0 as soon as it is known that the array is not a Martian array; continuing to analyze the array would be a waste of CPU cycles.

There should be exactly one for loop in your function

If you are programming in Java or C#, the function signature is

int isMartian(int[ ] a)

If you are programming in C or C++, the function signature is

int isMartian(int a[ ], int len) where len is the number of elements in the array a.

Examples

a is then function returns reason

{1, 3} 1 There is one 1 and zero 2s, hence the number of 1s is greater than the number of 2s. Also, no adjacent elements have the same value (1 does not equal 3)

{1, 2, 1, 2, 1, 2, 1, 2, 1} 1 There are five 1s and four 2s, hence the number of 1s is greater than the number of 2s. Also, no two adjacent elements have the same value.

{1, 3, 2} 0 There is one 1 and one 2, hence the number of 1s is not greater than the number of 2s.

{1, 3, 3, 2, 1} 0 There are two 3s adjacent to each other.

{1, 2, -18, -18, 1, 2} 0 The two -18s are adjacent to one another. Note that the number of 1s is not greater than than the number of 2s but your method should return 0 before determining that! (See the additional requirements above.)

{} 0 There are zero 1s and zero 2s hence the number of 1s is not greater than the number of 2s.

{1} 1 There is one 1 and zero 2s hence the number of 1s is greater than the number of 2s. Also since there is only one element, there cannot be adjacent elements with the same value.

{2} 0 There are zero 1s and one 2 hence the number of 1s is not greater than the number of 2s.

Hint: Make sure that your solution does not exceed the boundaries of the array!

private int isMartian(int[] a)

{

int sumOfOnes = 0;

int sumOfTwos = 0;

for (int i = 0; i < a.Length; i++)

{

if (i < a.Length - 1)

{

if (a[i] == a[i + 1])

return 0;

}

if (a[i] == 1)

sumOfOnes += 1;

else if (a[i] == 2)

sumOfTwos += 1;

}

if (sumOfOnes > sumOfTwos)

return 1;

else

return 0;

}