There are three questions on this test. You have two hours to complete it. Please do your own work. You are not allowed to use any methods or functions provided by the system unless explicitly stated in the question. In particular, you are not allowed to convert int to a String or vice-versa.

1. A **primeproduct** is a positive integer that is the product of exactly two primes greater than 1. For example, 22 is a primeproduct since 22 = 2 times 11 and both 2 and 11 are primes greater than 1. Write a function named *isPrimeProduct* with an integer parameter that returns 1 if the parameter is a primeproduct, otherwise it returns 0. Recall that a prime number is a positive integer with no factors other than 1 and itself.

You may assume that there exists a function named *isPrime(int m)* that returns 1 if its m is a prime number. You do not need to write *isPrime*. You are allowed to use this function.

The function signature
int isPrimeProduct(int n)

2. An array is called **balanced** if its even numbered elements (a[0], a[2], etc.) are even and its odd numbered elements (a[1], a[3], etc.) are odd.

Write a function named *isBalanced* that accepts an array of integers and returns 1 if the array is balanced, otherwise returns 0.

Examples: {2, 3, 6, 7} is balanced since a[0] and a[2] are even, a[1] and a[3] are odd. {6, 7, 2, 3, 12} is balanced since a[0], a[2] and a[4] are even, a[1] and a[3] are odd.
{7, 15, 2, 3} is not balanced since a[0] is odd.
{16, 6, 2, 3} is not balanced since a[1] is even.

If you are programming in Java or C#, the function signature is
int isBalanced(int[ ] a)

If you are programming in C or C++, the function signature is
int isBalanced(int a[ ], int len)
where *len* is the number of elements in the array.

3. An array with an odd number of elements is said to be **centered** if all elements (except the middle one) are strictly greater than the value of the middle element. Note that only arrays with an odd number of elements have a middle element. Write a function named *isCentered* that accepts an integer array and returns 1 if it is a centered array, otherwise it returns 0.

Examples: {5, 3, 3, 4, 5} is not a centered array (the middle element 3 is not strictly less than all other elements), {3, 2, 1, 4, 5} is centered (the middle element 1 is strictly less than all other elements), {3, 2, 1, 4, 1} is not centered (the middle element 1 is not strictly less than all other elements), {3, 2, 1, 1, 4, 6} is not centered (no middle element since array has even number of elements), {} is not centered (no middle element), {1} is centered (satisfies the condition vacuously).

If you are programming in Java or C#, the function signature is
int isCentered(int[ ] a)

If you are programming in C or C++, the function signature is
int isCentered(int a[ ], int len)
where *len* is the number of elements in the array.

There are three questions on this test. You have two hours to complete it. Please do your own work. You are not allowed to use any methods or functions provided by the system unless explicitly stated in the question. In particular, you are not allowed to convert int to a String or vice-versa.

1. Given a positive integer k, another positive integer n is said to have k-small factors if n can be written as a product

x, where x and y 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 is not less than 10.

Write a function *hasKSmallFactors* 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:
hasSmallFactors(7, 30) is true (since 5*6 = 30 and 5 < 7, 6 < 7).
hasSmallFactors(6, 14) is false (since the only way to factor 14 is 2*7 = 14 and 7 not less than 6)
hasSmallFactors(6, 30) is false (since 5*6 = 30, 6 not less than 6; 3 * 10 = 30, 10 not less than 6; 2 * 15 = 30, 15 not less than 6)

2. Write a function *fill* with signature

int[] fill(int[] arr, int k, int n)

which does the following: It returns an integer array arr2 of length n whose first k elements are the same as the first k elements of arr, and whose remaining elements consist of repeating blocks of the first k elements. You can assume array arr has at least k elements. The function should return null if either k or n is not positive.

Examples:
fill({1,2,3,5, 9, 12,-2,-1}, 3, 10) returns {1,2,3,1,2,3,1,2,3,1}.
fill({4, 2, -3, 12}, 1, 5) returns {4, 4, 4, 4, 4}.
fill({2, 6, 9, 0, -3}, 0, 4) returns null.

3. An array is said to be **hollow** if it contains 3 or more zeroes in the middle that are preceded and followed by the same number of non-zero elements. Write a function named *isHollow* that accepts an integer array and returns 1 if it is a hollow array, otherwise it returns 0

Examples: isHollow({1,2,4,0,0,0,3,4,5}) returns 1. isHollow ({1,2,0,0,0,3,4,5}) returns 0. isHollow ({1,2,4,9, 0,0,3,4, 5}) returns 0. isHollow ({1,2, 0,0, 3,4}) returns 0.

If you are programming in Java or C#, the function signature is
int isHollow(int[] a).

If you are C or C++ programmer
int isHollow(int[] a, int len)
where *len* is the number of elements in the array.

There are 3 questions on this test. You have 2 hours to finish it. Please do your own work. All you need to write is three functions. Please do not use any string functions. No sorting allowed. No additional arrays allowed. Try to write a simple, elegant and correct code.

1. Write a function named **minDistance** that returns the smallest distance between two factors of a number. For example, consider 13013 = 1*7*11*13. Its factors are 1, 7, 11, 13 and 13013. **minDistance**(13013) would return 2 because the smallest distance between any two factors is 2 (13 - 11 = 2). As another example, **minDistance** (8) would return 1 because the factors of 8 are 1, 2, 4, 8 and the smallest distance between any two factors is 1 (2 – 1 = 1).

The function signature is
int **minDistance**(int n)

2. A **wave array** is defined to an array which does **not** contain two even numbers or two odd numbers in adjacent locations. So {7, 2, 9, 10, 5}, {4, 11, 12, 1, 6}, {1, 0, 5} and {2} are all **wave arrays**. But {**2, 6**, 3, 4} is not a wave array because the even numbers 2 and 6 are adjacent to each other.

Write a function named *isWave* that returns 1 if its array argument is a Wave array, otherwise it returns 0.

If you are programming in Java or C#, the function signature is
int isWave (int [ ] a)

If you are programming in C or C++, the function signature is
int isWave (int a[ ], int len) where len is the number of elements in the array.

3. An array is defined to be a **Bean array** if it meets the following conditions
a. If it contains a 9 then it also contains a 13.
b. If it contains a 7 then it does **not** contain a 16.

So {1, 2, 3, **9**, 6, **13**} and {3, 4, 6, 7, 13, 15}, {1, 2, 3, 4, 10, 11, 12} and {3, 6, 9, 5, 7, 13, 6, 17} are Bean arrays. following arrays are **not** Bean arrays:
a. { 9, 6, 18} (contains a 9 but no 13)
b. {4, 7, 16} (contains both a 7 and a 16)

Write a function named isBean that returns 1 if its array argument is a Bean array, otherwise it returns 0.

If you are programming in Java or C#, the function signature is
int isBean (int[ ] a)

If you are programming in C or C++, the function signature is
int isBean (int a[ ], int len) where len is the number of elements in the array.

There are three questions on this test. You have two hours to complete it. Please do your own work. You are allowed to use any methods or functions provided by the system unless explicitly stated in the question. In particular you are not allowed to convert int to a String or vice-versa.

1. Write a function named **countDigit** that returns the number of times that a given digit appears in a positive number. For example countDigit(32121, 1) would return 2 because there are two 1s in 32121. Other examples:
countDigit(33331, 3) returns 4
countDigit(33331, 6) returns 0
countDigit(3, 3) returns 1

The function should return -1 if either argument is negative, so
countDigit(-543, 3) returns -1.

The function signature is
int countDigit(int n, int digit)

Hint: Use modulo base 10 and integer arithmetic to isolate the digits of the number.

2. A *Bunker array* is defined to be an array in which **at least one** odd number is immediately followed by a prime number. So {4, 9, 6, 7, 3} is a Bunker array because the odd number 7 is immediately followed by the prime number 3. But {4, 9, 6, 15, 21} is not a Bunker array because none of the odd numbers are immediately followed by a prime number.

Write a function named **isBunkerArray** that returns 1 if its array argument is a Bunker array, otherwise it returns 0.

If you are programming in Java or C#, the function signature is
int isBunkerArray(int [ ] a)

If you are programming in C or C++, the function signature is
int isBunkerArray(int a[ ], int len) where len is the number of elements in the array.

You may assume that there exists a function isPrime that returns 1 if it argument is a prime, otherwise it returns 0. **You do not have to write this function.**

A *Meera array* is defined to be an array such that for all values n in the array, the value $2*n$ is not in the array. So , 5, -2} is a Meera array because 3*2, 5*2 and -2*2 are not in the array. But {8, 3, 4} is not a Meera array because r n=4, $2*n$=8 is in the array.

rite a function named **isMeera** that returns 1 if its array argument is a Meera array. Otherwise it returns 0.

you are programming in Java or C#, the function signature is
int isMeera(int [ ] a)

you are programming in C or C++, the function signature is
int isMeera(int a[ ], int len) where len is the number of elements in the array.

ere are three questions on this test. You have two hours to complete it. Please do your own work. You are not lowed to use any methods or functions provided by the system unless explicitly stated in the question. In particular, u are not allowed to convert int to a String or vice-versa.

A **Meera number** is a number such that the number of nontrivial factors is a factor of the number. For example, 6 a Meera number because 6 has two nontrivial factors : 2 and 3. (A nontrivial factor is a factor other than 1 and the mber). Thus 6 has two nontrivial factors. Now, 2 is a factor of 6. Thus the number of nontrivial factors is a factor of Hence 6 is a Meera number. Another Meera number is 30 because 30 has 2, 3, 5, 6, 10, 15 as nontrivial factors. hus 30 has 6 nontrivial factors. Note that 6 is a factor of 30. So 30 is a Meera Number. However 21 is **not** a Meera mber. The nontrivial factors of 21 are 3 and 7. Thus the number of nontrivial factors is 2. Note that 2 is not a factor 21. Therefore, 21 is not a Meera number.

Vrite a function named *isMeera* that returns 1 if its integer argument is a Meera number, otherwise it returns 0.

he signature of the function is
int isMeera(int n)

A **Bunker array** is an array that contains the value 1 if and only if it contains a prime number. The array {7, 6, 10, } is a Bunker array because it contains a prime number (7) and also contains a 1. The array {7, 6, 10} is **not** a unker array because it contains a prime number (7) but does not contain a 1. The array {6, 10, 1} is **not** a Bunker rray because it contains a 1 but does not contain a prime number.

t is okay if a Bunker array contains more than one value 1 and more than one prime, so the array {3, 7, 1, 8, 1} is a unker array (3 and 7 are the primes).

Vrite a function named *isBunker* that returns 1 if its array argument is a Bunker array and returns 0 otherwise.

You may assume the existence of a function named *isPrime* that returns 1 if its argument is a prime and returns 0 otherwise. You do not have to write isPrime, you can just call it.

f you are programming in Java or C#, the function signature is
int isBunker(int [ ] a)

f you are programming in C or C++, the function signature is
int isBunker(int a[ ], int len) where len is the number of elements in the array.

3. A **Nice array** is defined to be an array where for every value n in the array, there is also an element n-1 or n+1 in the array.

For example, {2, 10, 9, 3} is a Nice array because
2 = 3-1
10 = 9+1
3 = 2 + 1
9 = 10 -1

Other Nice arrays include {2, 2, 3, 3, 3}, {1, 1, 1, 2, 1, 1} and {0, -1, 1}.

The array {3, 4, 5, 7} is **not** a Nice array because of the value 7 which requires that the array contains either the 6 (7-1) or 8 (7+1) but neither of these values are in the array.

Write a function named *isNice* that returns 1 if its array argument is a Nice array. Otherwise it returns a 0.

If you are programming in Java or C#, the function signature is
int isNice(int[ ] a)

If you are programming in C or C++, the function signature is
int isNice(int a[ ], int len) where len is the number of elements in the array.

There are 3 questions on this test. You have two hours to finish it. Please do your own work. All you need to write is two functions. Please do not use any string methods. No sorting allowed. No additional data structures including arrays allowed. Try to write a simple, elegant and correct code.

1. An integer is defined to be "continuous factored" if it can be expressed as the product of two or more continuous integers greater than 1.
Examples of "continuous factored" integers are:
6 = 2 * 3.
60 = 3 * 4 * 5
120 = 4 * 5 * 6
90 = 9*10
Examples of integers that are NOT "continuous factored" are: 99 = 9*11, 121=11*11, 2=2, 13=13

Write a function named ***isContinuousFactored(int n)*** that returns 1 if ***n*** is continuous factored and 0 otherwise.

2. Consider the prime number 11. Note that 13 is also a prime and 13 – 11 = 2. So, 11 and 13 are known as twin primes. Similarly, 29 and 31 are twin primes. So is 71 and 73. However, there are many primes for which there is no twin. Examples are 23, 67. A **twin array** is defined to an array which every prime that has a twin appear with a twin. Some examples are
{**3**, **5**, 8, 10, 27}, // 3 and 5 are twins and both are present
{**11**, 9, 12, **13**, 23}, // 11 and 13 are twins and both are present, 23 has no twin
{**5**, **3**, 14, 7, 18, 67}. // 3 and 5 are twins, 5 and 7 are twins, 67 has no twin

The following are NOT twin arrays:

{**13**, 14, 15, 3, 5} // 13 has a twin prime and it is missing in the array
{1, **17**, 8, 25, 67} // 17 has a twin prime and it is missing in the array

Write a function named *isTwin(int[ ] arr)* that returns 1 if its array argument is a Twin array, otherwise it returns 0.

3. Let us define two arrays as "set equal" if every element in one is also in the other and vice-versa. For example, any two of the following are equal to one another: {1, 9, 12}, {12, 1, 9}, {9, 1, 12, 1}, {1, 9, 12, 9, 12, 1, 9}. Note that {1, 7, 8} is not set equal to {1, 7, 1} or {1, 7, 6}.

Write a function named *isSetEqual(int[ ] a, int[ ] b)* that returns 1 if its array arguments are set equal, otherwise it returns 0.

There are 3 questions on this test. You have two hours to finish it. Please do your own work. All you need to write is three functions. Please do not use any string methods. No sorting allowed. No additional data structures including arrays allowed. Try to write a simple, elegant and correct code.

An integer is defined to be a **Smart** number if it is an element in the infinite sequence 1, 2, 4, 7, 11, 16, ... Note that 1 + 1 = 2, 2 + 2 = 4, 4 + 3 = 7, 7 + 4 = 11, 11 + 5 = 16 so for k = 1, the kth element of the sequence is equal to the k-1th element + k-1. For example, for k = 6, 16 is the kth element and is equal to 11 (the k-1th element) + 5 (k-1).

Write a function named *isSmart* that returns 1 if its argument is a Smart number, otherwise it returns 0. So isSmart(11) returns 1, isSmart(12) returns 1 and isSmart(8) returns 0.

The function signature is
int isSmart(int n)

An array is defined to be a **Nice** array if the sum of the primes in the array is equal to the first element of the array. If there are no primes in the array, the first element must be 0. So {21, 3, 7, 9, 11, 4, 6} is a Nice array because 3, 7, 11 are the primes in the array and they sum to 21 which is the first element of the array. {13, 4, 4, 4, 4} is also a Nice array because the sum of the primes is 13 which is also the first element. Other Nice arrays are {10, 5, 5}, {0, 6, 8, 20} and {4}. {8, 5, 5, 5, 3} is **not** a Nice array because the sum of the primes is 5+5+3 = 13 but the first element of the array is 8. Note that -5 is not a prime because prime numbers are positive.

Write a function named *isNiceArray* that returns 1 if its integer array argument is a Nice array. Otherwise it returns 0.

The function signature is
int isNiceArray (int[ ] a)

You may assume that a function named *isPrime* exists that returns 1 if its int argument is a prime, otherwise it returns 0. **You do **not** have to write this function!** You just have to call it.

3. An array is defined to be **complete** if all its elements are greater than 0 and all even numbers that are less than the maximum even number are in the array.

For example {2, 3, 2, 4, 11, 6, 10, 9, 8} is complete because
a. all its elements are greater than 0
b. the maximum even integer is 10
c. all even numbers that are less than 10 (2, 4, 6, 8) are in the array.

But {2, 3, 3, 6} is **not** complete because the even number 4 is missing. {2, -3, 4, 3, 6} is **not** complete because it contains a negative number.

Write a function named *isComplete* that returns 1 if its array argument is a complete array. Otherwise it returns 0.

The function signature is
int isComplete (int[ ] a)

There are 3 questions on this test. You have two hours to finish it. Please do your own work. All you need to write is three functions. Please do not use any string methods. No sorting allowed. No additional data structures including arrays allowed. Try to write a simple, elegant and correct code.

Two integers are defined to be ***factor equal***, if they have the same number of factors. For example, integers 10 and 33 are factor equal because, 10 has four factors: 1, 2, 5, 10 and 33 also has four factors: 1, 3, 11, 33. On the other hand, 9 and 10 are not factor equal since 9 has only three factors: 1, 3, 9 and 10 has four factors: 1, 2, 5, 10.
Write a function named *factorEqual(int n, int m)* that returns 1 if *n* and *m* are factor equal and 0 otherwise.
The signature of the function is
int factorEqual(int n, int m)

2. Define a **Meera array** to be an array *a* if it satisfies two conditions:
(a) *a[i]* is less than *i* for *i = 0* to *a.length-1*.
(b) sum of all elements of *a* is 0.

For example, {-4, 0, 1, 0, 2, 1} is a Meera array because
-4 = a[0] - 0
0 = a[1] - 1
1 = a[2] - 2
0 = a[3] - 3
2 = a[4] - 4
1 = a[5] - 5

and -4 + 0 + 1 + 0 + 2 + 1 = 0

{-8, 0, 0, 8, 0} is not a Meera array because a[3] is 8 which is not less than 3. Thus condition (a) fails. {-8, 0, ... is not a Meera array because -8 + 2 = -6 not equal to zero. Thus condition (b) fails.

Write a function named *isMeera* that returns 1 if its array argument is a Meera array. Otherwise it returns 0.

If you are programming in Java or C#, the function signature is
int isMeera (int[ ] a)

If you are programming in C or C++, the function signature is
int isMeera (int a[ ], int len) where *len* is the number of elements in the array.

3. Define a **Triple array** to be an array where every value occurs exactly three times.

For example, {3, 1, 2, 1, 3, 1, 3, 2, 2} is a Triple array.

The following arrays are **not** Triple arrays
{2, 5, 2, 5, 5, 2, 5} (5 occurs four times instead of three times)
{3, 1, 1, 1} (3 occurs once instead of three times)

Write a function named *isTriple* that returns 1 if its array argument is a Triple array. Otherwise it returns 0.

If you are programming in Java or C#, the function signature is
int isTriple (int[ ] a)

If you are programming in C or C++, the function signature is
int isTriple (int a[ ], int len) where len is the number of elements in the array.

There are 3 questions on this test. You have two hours to finish it. Please do your own work. All you need t write is three functions. Please do not use any string methods. No sorting allowed. No additional data stru including arrays allowed. Try to write a simple, elegant and correct code.

1. A **Fibonacci number** is a number in the sequence 1, 1, 2, 3, 5, 8, 13, 21,…. Note that first two Fibonacci num are 1 and any Fibonacci number other than the first two is the sum of the previous two Fibonacci numbers. For example, 2 = 1 + 1, 3 = 2 + 1, 5 = 3 + 2 and so on.

Write a function named *isFibonacci* that returns 1 if its integer argument is a Fibonacci number, otherwise it retu

The signature of the function is
int isFibonacci (int n)

2. A **Meera array** is an array that contains the value 0 if and only if it contains a prime number. The array {7, 6, 1} is a Meera array because it contains a prime number (7) and also contains a 0. The array {6, 10, 1} is a Meera because it contains no prime number and also contains no 0.

array because it contains a prime number (7) but does not contain a 0. The array
not a Meera array because it contains a 0 but does not contain a prime number.

if a Meera array contains more than one value 0 and more than one prime, so the array {3, 7, 0, 8, 0, 5} is a
array (3, 5 and 7 are the primes and there are two zeros.).

function named *isMeera* that returns 1 if its array argument is a Meera array and returns 0 otherwise.

assume the existence of a function named *isPrime* that returns 1 if its argument is a prime and returns 0
wise. You do not have to write isPrime, you can just call it.
are programming in Java or C#, the function signature is
Meera(int [ ] a)
are are programming in C or C++, the function signature is
isMeera(int a[ ], int len) where *len* is the number of elements in the array.
Bean array is defined to be an array where for every value *n* in the array, there is also an element n-1 or n+1 in
array.
example, {2, 10, 9, 3} is a Bean array because

+1
-1
+1
10-1

Bean arrays include {2, 2, 3, 3, 3}, {1, 1, 1, 2, 1, 1} and {0, -1, 1}.
array {3, 4, 5, 7} is **not** a Bean array because of the value 7 which requires that the array contains either the value
-1) or 8 (7+1) but neither of these values are in the array.

ite a function named *isBean* that returns 1 if its array argument is a *Bean* array. Otherwise it returns a 0.
ou are programming in Java or C#, the function signature is
nt isBean(int[ ] a)

you are programming in C or C++, the function signature is
int isBean(int a[ ], int len) where *len* is the number of elements in the array.

here are 3 questions on this test. You have two hours to finish it. Please do your own work. All you need to rite is three functions. Please do not write main method. If you write, you are just wasting your time! Please o not use any string methods. No sorting allowed. No additional data structures including arrays allowed. Try o write a simple, elegant and correct code.

1. A **fancy number** is a number in the sequence 1, 1, 5, 17, 61, … .Note that first two fancy numbers are 1 and any fancy number other than the first two is sum of the three times previous one and two times the one before that. See below:

1,
1,
3*1+2*1 = 5
3*5+2*1 = 17
3*17+2*5 = 61

Write a function named *isFancy* that returns 1 if its integer argument is a Fancy number, otherwise it returns 0. The signature of the function is

int isFancy(int n)

2. A **Meera array** is an array that contains the value 1 if and only if it contains 9. The array {7, 9, 0, 10, 1} is a Meera array because it contains 1 and 9. The array {6, 10, 8} is a Meera array because it contains no 1 and also contains no 9. The array {7, 6, 1} is **not** a Meera array because it contains 1 but does not contain a 9. The array {9, 10, 0} is **not** a Meera array because it contains a 9 but does not contain 1.
It is okay if a Meera array contains more than one value 1 and more than one 9, so the array {1, 1, 0, 8, 0, 9, 9, 1} is a Meera array.

Write a function named *isMeera* that returns 1 if its array argument is a Meera array and returns 0 otherwise.
If you are programming in Java or C#, the function signature is
int isMeera(int [ ] a)
If you are are programming in C or C++, the function signature is
int isMeera(int a[ ], int len) where *len* is the number of elements in the array.

3. A **Bean array** is defined to be an integer array where for every value *n* in the array, there is also an element 2n, 2n+1 or n/2 in the array.
For example, {4, 9, 8} is a Bean array because
For 4, 8 is present; for 9, 4 is present; for 8, 4 is present.
Other Bean arrays include {2, 2, 5, 11, 23}, {7, 7, 3, 6} and {0}.
The array {3, 8, 4} is **not** a Bean array because of the value 3 which requires that the array contains either the value 7 or 1 and none of these values are in the array.
Write a function named *isBean* that returns 1 if its array argument is a *Bean* array. Otherwise it returns a 0.

If you are programming in Java or C#, the function signature is
int isBean(int[ ] a)

If you are programming in C or C++, the function signature is
int isBean(int a[ ], int len) where *len* is the number of elements in the array.

There are 3 questions on this test. You have two hours to finish it. Please do your own work. All you need to write is three functions. Please do not write main method. If you write, you are just wasting your time! Please do not use any string methods. No sorting allowed. No additional data structures including arrays allowed. To write a simple, elegant and correct code.

1. An integer is defined to be a **Bunker** number if it is an element in the infinite sequence 1, 2, 4, 7, 11, 16, 22, ...
Note that 2-1=1, 4-2=2, 7-4=3, 11-7=4, 16-11=5 so for k>1, the kth element of the sequence is equal to the k-1th element + k-1. E.G., for k=6, 16 is the kth element and is equal to 11 (the k-1th element) + 5 (k-1).
Write function named *isBunker* that returns 1 if its argument is a Bunker number, otherwise it returns 0. So *isBunker*(11) returns 1, *isBunker*(22) returns 1 and *isBunker*(8) returns 0 .
The function signature is
int isBunker (int n)

2. Define a **Dual array** to be an array where every value occurs exactly twice.
For example, {1, 2, 1, 3, 3, 2} is a dual array.
The following arrays are **not** Dual arrays
{2, 5, 2, 5, 5} (5 occurs three times instead of two times)
{3, 1, 1, 2, 2} (3 occurs once instead of two times)
Write a function named *isDual* that returns 1 if its array argument is a Dual array. 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.

3. An array is defined to be a **Filter array** if it meets the following conditions
a. If it contains 9 then it also contains 11.
b. If it contains 7 then it does **not** contain 13.
So {1, 2, 3, **9**, 6, **11**} and {3, 4, 6, 7, 14, 16}, {1, 2, 3, 4, 10, 11, 13} and {3, 6, 5, 5, 13, 6, 13} are Filter arrays. The following arrays are **not** Filter arrays: {9, 6, 18} (contains 9 but no 11), {4, 7, 13} (contains both 7 and 13)
Write a function named *isFilter* that returns 1 if its array argument is a Filter array, otherwise it returns 0.
If you are programming in Java or C#, the function signature is
int isFilter(int[ ] a)
If you are programming in C or C++, the function signature is
int isFilter(int a[ ], int len) where len is the number of elements in the array.

There are 3 questions on this test. You have two hours to finish it. Please do your own work. All you need to write is three functions. Please do not write main method. If you write, you are just wasting your time! Please do not use any string methods. No sorting allowed. No additional data structures including arrays allowed. Try to write a simple, elegant and correct code.

Question 1. An array is called ***balanced*** if its even numbered elements (a[0], a[2], etc.) are even and its odd numbered elements (a[1], a[3], etc.) are odd. Write a function named *isBalanced* that accepts an array of integers and returns 1 if the array is balanced, otherwise it returns 0. Examples: {2, 3, 6, 7} is balanced since a[0] and a[2] are even, a[1] and a[3] are odd. {6, 7, 2, 3, 12} is balanced since a[0], a[2] and a[4] are even, a[1] and a[3] are odd. {7, 15, 2, 3} is not balanced since a[0] is odd. {16, 6, 2, 3} is not balanced since a[1] is even.
If you are programming in Java or C#, the function signature is

int isBalanced(int[ ] a)

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

int isBalanced(int a[ ], int len)

where *len* is the number of elements in the array.

Question 2. An array is defined to be ***odd-heavy*** if it contains at least one odd element and every odd element is greater than every even element. So {11, 4, 9, 2, 8} is odd-heavy because the two odd elements (11 and 9) are greater than all the even elements. And {11, 4, 9, 2, 3, 10} is not odd-heavy because the even element 10 is greater than the odd element 9. Write a function called *isOddHeavy* that accepts an integer array and returns 1 if the array is odd-heavy; otherwise it returns 0. Some other examples: {1} is odd-heavy. {2} is not odd-heavy. {1, 1, 1, 1} is odd-heavy, {2, 4, 6, 8, 11} is odd-heavy, {-2, -4, -6, -8, -11} is not odd-heavy.
If you are programming in Java or C#, the function signature is

int isOddHeavy(int[ ] a)

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

int isOddHeavy(int a[ ], int len)

where *len* is the number of elements in the array.

Question 3. A ***normal*** number is defined to be one that has no odd factors, except for 1 and possibly itself. Write a method named *isNormal* that returns 1 if its integer argument is normal, otherwise it returns 0. The function signature is

int isNormal(int n)

Examples: 1, 2, 3, 4, 5, 7, 8 are normal numbers. 6 and 9 are not normal numbers since 3 is an odd factor. 10 is not a normal number since 5 is an odd factor.

There are 3 questions on this test. You have two hours to finish it. Please do your own work. All you need to write is three functions. Please do not write main method. If you write, you are just wasting your time! Please do not use any string methods. No sorting allowed. No additional data structures including arrays allowed. Try to write a simple, elegant and correct code.

Question 1. An array with an odd number of elements is said to be ***centered*** if all elements (except the middle one) are strictly greater than the value of the middle element. Note that only arrays with an odd number of elements have a middle element. Write a function named *isCentered* that accepts an integer array and returns 1 if it is a centered array, otherwise it returns 0. Examples: {1, 2, 3, 4, 5} is not a centered array (the middle element 3 is not strictly less than all other elements), {3, 2, 1, 4, 5} is centered (the middle element 1 is strictly less than all other elements), {3, 2, 1, 1, 4, 6} is not centered (no 1} is not centered (the middle element 1 is not strictly less than all other elements), {} is not centered (no middle element), {1} is centered (satisfies the condition vacuously).
If you are programming in Java or C#, the function signature is

int isCentered(int[ ] a)

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

int isCentered(int a[ ], int len)

where *len* is the number of elements in the array.

Question 2. 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. Examples: {1, 2, 3, 0} is a dual array (because 1+2 = 3+0 = 3), {1, 2, 2, 1, 3, 0} is a dual array (because 1+2 = 2+1 = 3+0 = 3), {1, 1, 2, 2}<td> is not a dual array (because 1+1 is not equal t

array.

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.

Question 3. A non-empty array of length n is called an ***array of all possibilities***, if it contains all numbers between 0 and n - 1 inclusive. Write a method named *isAllPossibilities* that accepts an integer array and returns 1 if it is an array of all possibilities, otherwise it returns 0. Examples {1, 2, 0, 3} is an array of all possibilities, {1, 2, 4, 3} is not an array of all possibilities, (because 0 not included and 4 is), {1, 2, 3} is not an array of all possibilities, (because 1 is not included), {0} is an array of all possibilities, {} is not an array of all possibilities (because array is empty).

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

```
int isAllPossibilities(int[ ] a)
```

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

```
int isAllPossibilities(int a[ ], int len)
```

where *len* is the number of elements in the array.

There are 3 questions on this test. You have two hours to finish it. Please do your own work. All you need to write is three functions. Please do not use any string methods. No sorting allowed. No additional data structures including arrays allowed. Try to write a simple, elegant and correct code.

1. Write a function named **factorTwoCount** that returns the number of times that 2 divides the argument.

For example, factorTwoCount(48) returns 4 because
48 / 2 = 24
24 / 2 = 12
12 / 2 = 6
6 / 2 = 3
2 does not divide 3 evenly.

Another example: factorTwoCount(27) returns 0 because 2 does not divide 27.

The function signature is
int factorTwoCount(int n);

2. A **Daphne array** is defined to be an array that contains at least one odd number and begins and ends with the same number of even numbers.

So {**4, 8, 6,** 3, 2, 9, 8,11, 8, 13, **12, 12, 6**} is a Daphne array because it begins with three even numbers and ends with three even numbers and it contains at least one odd number

The array {2, 4, 6, 8, 6} is not a Daphne array because it does not contain an odd number.

The array {**2, 8,** 7, **10, -4, 6**} is not a Daphne array because it begins with two even numbers but ends with three even numbers.

Write a function named *isDaphne* that returns 1 if its array argument is a Daphne array. Otherwise, it returns 0.

If you are writing in Java or C#, the function signature is
int isDaphne (int[ ] a)

If you are writing in C or C++, the function signature is
int isDaphne (int a[ ], int len) where len is the number of elements in the array.

3. Write a function called **goodSpread** that returns 1 if no value in its array argument occurs more than 3 times

If you are writing in C or C++, the function signature is
int goodSpread (int a[ ], int len) where len is the number of elements in the array.

There are three questions on this test. You have two hours to complete it. Please do your own work.

1. Write a function named *sumDigits* that sums the digits of its integer argument. For example sumDigits(311) returns 5, sumDigits(-6543) returns 18 and sumDigits(0) returns 0.

The signature of the function is
int sumDigits (int n)

2. Define a **Meera array** to be an array where a[n] is less than n for n = 0 to a.length-1.

For example, {-4, 0, 1, 0, 2} is a Meera array because
a[0] < 0
a[1] < 1
a[2] < 2
a[3] < 3
a[4] < 4

{-1, 0, 0, 8, 0} is not a Meera array because a[3] is 8 which is not less than 3.

Write a function named *isMeera* that returns 1 if its array argument is a Meera array. Otherwise it returns 0.

If you are programming in Java or C#, the function signature is
int isMeera (int[ ] a)

If you are programming in C or C++, the function signature is
int isMeera (int a[ ], int len) where len is the number of elements in the array.

3. Define a **Dual array** to be an array where every value occurs exactly twice.
For example, {1, 2, 1, 3, 3, 2} is a dual array.
The following arrays are **not** Dual arrays
{2, 5, 2, 5, 5} (5 occurs three times instead of two times)
{3, 1, 1, 2, 2} (3 occurs once instead of two times)
Write a function named *isDual* that returns 1 if its array argument is a Dual array. 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.
Hint: you need a nested loop.
There are three questions on this test. You have two hours to complete it. Please do your own work.

1. An integer is defined to be a **Guthrie** number if it is an element in the infinite sequence 1, 2, 4, 7, 11, 16... where that 2-1=1, 4-2=2, 7-4=3, 11-7=4, 16-11=5 so for k>1, the kth element of the sequence is equal to the k-1th element + k-1. E.G., for k=6, 16 is the kth element and is equal to 11 (the k-1th element) + k-1.
Write function named *isGuthrie* that returns 1 if its argument is a Guthrie number, otherwise it returns 0. So isGuthrie(11) returns 1, is Guthrie(22) returns 1 and isGuthrie(8) returns 0

The function signature is
int isGuthrie (int n)

2. An array is defined to be a **Bean** array if the sum of the primes in the array is equal to the first element of the array. If there are no primes in the array, the first element must be 0. So {21, 3, 7, 9, 11 4, 6} is a Bean array because

...are programming in C or C++, the function signature is

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

...are three questions on this test. You have two hours to complete it. Please do your own work.

Pascal number is a number that is the sum of the integers from 1 to j for some j. For example 6 is a Pascal ...se because 6 = 1 + 2 + 3. Here j is 3. Another Pascal number is 15 because 15 = 1 + 2 + 3 + 4 + 5. An example of ...ber that is not a Pascal number is 7 because it falls between the Pascal numbers 6 and 10.

...a function named *isPascal* that returns 1 if its integer argument is a Pascal number, otherwise it returns 0.

...signature of the function is

```
int isPascal (int n)
```

A Meera array is an array that contains the value 1 if and only if it contains a prime number. The array {7, 6, 10, ...a Meera array because it contains a prime number (7) and also contains a 1. The array {7, 6, 10} is **not** a ...ra array because it contains a prime number (7) but does not contain a 1. The array {6, 10, 1} is **not** a Meera array ...use it contains a 1 but does not contain a prime number.

...s okay if a Meera array contains more than one value 1 and more than one prime, so the array {3, 7, 1, 8, 1} is a ...era array (3 and 7 are the primes).

...rite a function named *isMeera* that returns 1 if its array argument is a Meera array and returns 0 otherwise.

...ou may assume the existence of a function named *isPrime* that returns 1 if its argument is a prime and returns 0 ...herwise. You do not have to write isPrime, you can just call it.

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

```
int isMeera (int [ ] a)
```

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

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

3. A **Suff array** is defined to be an array where for every value n in the array, there is also an element n-1 or n+1 in ...e array.

For example, {2, 10, 9, 3} is a Suff array because

2 = 3-1

10 = 9+1

3 = 2 + 1

9 = 10 -1

Other Suff arrays include {2, 2, 3, 3, 3}, {1, 1, 1, 2, 1, 1} and {0, -1, 1}.

The array {3, 4, 5, 7} is **not** a Suff array because of the value 7 which requires that the array contains either the value 6 (7-1) or 8 (7+1) but neither of these values are in the array.

Write a function named *isSuff* that returns 1 if its array argument is a Suff array. Otherwise it returns a 0.

...the following. It returns an integer array arr2 of length n whose first k elements are the same as the first k ... of arr, and whose remaining elements consist of repeating blocks of the first k elements. You can assume ... has at least k elements. The function should return null if either k or n is not positive.

...Fill({2, 5, 5, 9, 12, -2, -1}, 3, 10) returns {2, 5, 5, 2, 5, 5, 2, 5, 5, 2} Fill({4, 2, 4, 1}, 1, 6) returns {4, 4, 4, 4, 4, 4}
...({2, 5, 2, 0, -3}, 0, 4) returns null.

...estion 2. Write a function sumIsPower with signature

```
boolean sumIsPower(int[] arr)
```

...n outputs true if the sum of the elements in the input array arr is a power of 2, false otherwise. Recall that the ...ers of 2 are 1, 2, 4, 8, 16, and so on. In general a number is a power of 2 if and only if it is of the form 2^n for some ...negative integer n. You may assume (without verifying in your code) that all elements in the array are positive ...bers. If the input array arr is null, the return value should be false.

...amples: sumIsPower({8,8,8,8}) is true since 8 + 8 + 8 + 8 = 32 = 2^5. sumIsPower({8,8,8}) is false, since 8 + 8 + 8 ... not a power of 2.

...estion 3. An array is said to be hollow if it contains 3 or more zeros in the middle that are preceded and followed ... the same number of non-zero elements. Write a function named isHollow that accepts an integer array and returns ... it is a hollow array, otherwise it returns 0. The function signature is

```
...sHollow(int[ ] a).
```

...amples: isHollow({1,2,4,0,0,0,3,4,5}) returns true. isHollow ({1,2,0,0,0,3,4,5}) returns false. isHollow ({1, 2, 4, 0, ...,3,4, 5}) returns false. isHollow ({1,2, 0,0, 3,4}) returns false.

...y 2nd 2015

There are 3 questions on this test. You have two hours to finish it. Please do your own work. All you need to write is three functions. Please do not use any string methods. No sorting allowed. No additional data structures including arrays allowed. Try to write a simple, elegant and correct code.

A **Fibonacci number** is a number in the sequence 1, 1, 2, 3, 5, 8, 13, 21,.... Note that first two Fibonacci numbers are 1 and any Fibonacci number other than the first two is the sum ofthe previous two Fibonacci numbers. For example, 2 = 1 + 1, 3 = 2 + 1, 5 = 3 + 2 and so on.

Write a function named *isFibonacci* that returns 1 if its integer argument is a Fibonacci number, otherwise it returns 0.

The signature of the function is

```
int isFibonacci (int n)
```

A **Meera array** is an array that contains the value 0 if and only if it contains a prime number. The array {7, 6, 0, 10, 1} is a Meera array because it contains a prime number (7) and also contains a 0. The array {6, 10, 1} is a Meera array because it contains no prime number and also contains no 0.

The array {7, 6, 10} is **not** a Meera array because it contains a prime number (7) but does not contain a 0. The array {6, 10, 0} is **not** a Meera array because it contains a 0 but does not contain a prime number.

It is okay if a Meera array contains more than one value 0 and more than one prime, so the array {3, 7, 0, 8, 0, 5} is a Meera array (3, 5 and 7 are the primes and there are two zeros.).

Write a function named *isMeera* that returns 1 if its array argument is a Meera array and returns 0 otherwise.

You may assume the existence of a function named *isPrime* that returns 1 if its argument is a prime and 0 otherwise. You do not have to write *isPrime*, you can just call it.

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

int isMersenne(int n)

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

int isMersenne(int n) where *len* is the number of elements in the array.

1. A **Bean array** is defined to be an array where for every value *n* in the array, there is also an element n-1 in the array.

For example {2, 10, 9, 3} is a Bean array because:

2 has 1

10 has 9

3 has 2

9 has 10

Other Bean arrays include {2, 2, 3, 3, 3}, {1, 1, 1, 2, 1, 1} and {0, -1, 1}.

The array {3, 4, 5, 7} is **not** a Bean array because of the value 7 which requires that the array contains either the value 6 (7 - 1) or 8 (7 + 1) but neither of these values are in the array.

Write a function named *isBean* that returns 1 if its array argument is a *Bean* array. Otherwise it returns a 0.

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

int isBean(int[] a)

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

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

May 16, 2015

There are three questions on this test. You have two hours to complete it. Please do your own work.

1. A **Riley** number is an integer whose digits are all even. For example 2426 is a Riley number but 3224 is not.

Write a function named *isRiley* that returns 1 if its integer argument is a Riley number otherwise it returns 0.

The function signature is

int isRiley (int n)

2. Write a function named *lastEven* that returns the index of the last even value in its array argument. For example, lastEven will return 3 if the array is {3, 2, 5, 6, 7}, because that is the index of 6 which is the last even value in the array.

the array has no even numbers, the function should return -1

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

int lastEven (int[ ] a)

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

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

1. Write a function named *countMax* that returns the number of times that the max value occurs in the array. For example, countMax would return 2 if the array is {6, 3, 1, 3, 4, 3, 6, 5} because 6 occurs 2 times in the array.

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

int countMax (int[ ] a)

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

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

May 30, 2015 Test

There are three questions on this test. You have two hours to finish it. Please do your own work.

1. An integer is defined to be an **even subset** of another integer *n* if every even factor of *m* is also a factor of *n*. For example 18 is an even subset of 12 because the even factors of 18 are 2 and 6 and these are both factors of 12. But 18 is not an even subset of 32 because 6 is not a factor of 32.

Write a function with signature **int isEvenSubset(int m, int n)** that returns 1 if m is an even subset of n, otherwise it returns 0.

2. A **twinoid** is defined to be an array that has exactly two even values that are adjacent to one another. For example {3, 3, 2, 6, 7} is a twinoid array because it has exactly two even values (2 and 6) and they are adjacent to one another. The following arrays are not twinoid arrays.

{3, 3, 2, 6, 6, 7} because it has three even values.

{3, 3, 2, 7, 6, 7} because the even values are not adjacent to one another

{3, 8, 5, 7, 3} because it has only one even value.

Write a function named **isTwinoid** that returns 1 if its array argument is a twinoid array. Otherwise it returns 0.

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

int isTwinoid (int [ ] a);

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

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

A **balanced** array is defined to be an array where for every value it in the array, it also is in the array. For example, {2, 5, 2, -5} is a balanced array. So is {-2, -5, 2, 5}. But {-5, 2, -2} is not because -5 is not in the array.

Write a function named isBalanced that returns 1 if its array argument is a balanced array. Otherwise it returns 0.

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

int isBalanced(int[] a)

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

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

June 13, 2018

This exam is two hours long and contains three questions. Please indent your code so it is easy for the grader to read.

1. Write a method named **geekXponent(n, p)** that returns the largest exponent x such that p^x evenly divides n. If n <= 1 the method should return -1.

For example, geekXponent(162, 3) returns 4 because 162 = $2^1 * 3^4$, therefore the value of x here is 4.

The method signature is
int geekXponent(int n, int p)

Examples:

If n is	and p is	return	Because
27	3	3	3^3 divides 27 evenly but 3^4 does not.
28	3	0	3^0 divides 28 evenly but 3^1 does not.
280	-7	1	-7^1 divides 280 evenly but -7^2 does not.
-250	5	3	5^3 divides -250 evenly but 5^4 does not.
18	1	-1	If p <=1 the function returns -1.
-128	4	3	4^3 divides 128 evenly but 4^4 does not.

2. Define an array to be a 121 array if all its elements are either 1 or 2 and it begins with one or more 1s followed by one or more 2s and then ends with the same number of 1s that it begins with. Write a method named **is121Array** that returns 1 if its array argument is a 121 array, otherwise, it returns 0.

If you are programming in Java or C#, the function signature is
int is121Array(int[] a)

If you are programming in C or C++, the function signature is
int is121Array(int a[], int len) where len is the number of elements in the array a.

Examples

a is	then function returns	reason
{1, 2, 1}	1	because the same number of 1s are at the beginning

	and end of the array and there is at least one 2 in between them.
1	because the same number of 1s are at the beginning and end of the array and there is at least one 2 in between them.
0	Because the number of 1s at the end does not equal the number of 1s at the beginning.
0	Because the middle does not contain only 2s.
0	Because the array contains a number other than 1 and 2.
0	Because the array does not contain any 2s
0	Because the first element of the array is not a 1.
0	Because the last element of the array is not a 1.
0	Because there are no 1s in the array.

...is defined to be **maxmin equal** if it contains at least two **different** elements and the number of times the ...value occur is the same as the number of times the minimum value occur. So {11, 4, 9, 11, 8, 5 , 4, 10} is ...equal, because the max value 11 and min value 4 both appear two times in the array.

...function called *isMaxMinEqual* that accepts an integer array and returns 1 if the array is **maxmin equal**; ...se it returns 0.

...re programming in Java or C#, the function signature is
...int *isMaxMinEqual*(int[ ] a)

...re programming in C or C++, the function signature is
...int *isMaxMinEqual*(int a[ ], int len) where len is the number of elements in the array

...other examples:

...if the input array is	*isMaxMinEqual* **should return**
	0 (array must have at least two *different* elements)
	0 (array must have at least two *different* elements)
...1, 1, 1, 1, 11	0 (array must have at least two *different* elements)
...4, 6, 8, 11}	1 (Both max value (11) and min value 2 appear exactly one time)
...-2, -6, -8, -11}	1 (Both max value (-2) and min value -11 appear exactly one time)

...ent 2nd Test 2015

Do not use any string methods or string operations. *No String properties if your program is in C#.* No sorting allowed. No additional data structures including arraysallowed. Try to write simple, elegant and correct code. You will be graded both on correctness and efficiency.

An integer array is said to be *oddSpaced*, if the difference between the largest value and the smallest value is an odd number. Write a function *isOddSpaced(int[] a)* that will return 1 if it is*oddSpaced* and 0 otherwise. If array has less than two elements, function will return 0. If you are programming in C or C++, the function signature is:

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

Examples

Array	Largest value	Smallest value	Difference	Return value
{100, 19, 131, 140}	140	19	140 - 19 = 121	1
{200, 1, 151, 160}	200	1	200 - 1 = 199	1
{200, 10, 151, 160}	200	10	200 - 10 = 190	0
{100, 19, -131, -140}	100	-140	100 - (-140) = 240	0
{80, -56, 11, -81}	80	-81	80 - (-81) = 161	1

2. An *Super array* is defined to be an array in which each element is greater than sum of all elements before that. See examples below:

{2, 3, 6, 13} is a *Super* array. Note that 2 < 3, 2+3 < 6, 2 + 3 + 6 < 13.

{2, 3, 5, 11} is a NOT a *Super* array. Note that 2 + 3 not less than 5.

Write a function named *isSuper* that returns 1 if its array argument is a *isSuper* array, otherwise it returns 0.

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

int *isSuper* (int [ ] a)

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

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

3. An *isSym* (even/odd Symmetric) *array* is defined to be an array in which even numbers and odd numbers appear in the same order from "*both directions*". You can assume array has at least one element. See examples below:

{2, 7, 9, 10, 11, 5, 8} is a *isSym* array.

Note that from left to right or right to left we have even, odd, odd, even, odd, odd, even.

{9, 8, 7, 13, 14, 17} is a *isSym* array.

Note that from left to right or right to left we have {odd, even, odd, odd, even, odd}.

However, {2, 7, 8, 9, 11, 13, 10} is not a *isSym* array.

From left to right we have {even, odd, even, odd, odd, odd, even}.

From right to left we have {even, odd, odd, odd, even, odd, even},

whichis not the same.

Write a function named *isSym* that returns 1 if its array argument is a *isSym* array, otherwise it returns 0.

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

int *isSym* (int [ ] a)

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

[illegible] 20 : 5

Do not use any string methods or string operations. *No String properties if your program is in C#. No strings allowed.* No additional data structures including arrays allowed. Try to write simple, elegant and correct code. You will be graded both on correctness and efficiency.

An integer array is said to be *evenSpaced*, if the difference between the largest value and the smallest value is an even number. Write a function *isEvenSpaced(int[] a)* that will return 1 if it is *evenSpaced* and 0 otherwise. If array has less than two elements, function will return 0. If you are programming in C or C++, the function signature is

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

Examples

Array	Largest value	Smallest value	Difference	Return value
{10, 19, 131, 140}	140	19	140 - 19 = 121	0
{200, 1, 151, 160}	200	1	200 - 1 = 199	0
{200, 10, 151, 160}	200	10	200 - 10 = 190	1
{100, 19, -131, -140}	100	-140	100 - (-140) = 240	1
{80, -56, 11, -81}	80	-81	-80 - 80 = -161	0

2. An *Sub array* is defined to be an array in which each element is greater than sum of all elements **after** that. See examples below:

{13, 6, 3, 2} is a *Sub* array. Note that 13 > 2 + 3 + 6, 6 > 3 + 2, 3 > 2.

{11, 5, 3, 2} is a NOT a *Sub* array. None that 5 is not greater than 3 + 2.

Write a function named *isSub* that returns 1 if its array argument is a *Sub* array, otherwise it returns 0.

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

int *isSub* (int [ ] a)

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

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

3. An *isSym* (even/odd Symmetric) *array* is defined to be an array in which even numbers and odd numbers appear in the same order from "*both directions*". You can assume array has at least one element. See examples below:

{2, 7, 9, 10, 11, 5, 8} is a *isSym* array.

Note that from left to right or right to left we have even, odd, odd, even, odd, odd, even.

{9, 8, 7, 13, 14, 17} is a *isSym* array.

Note that from left to right or right to left we have {odd, even, odd, odd, even, odd}.

However, {2, 7, 8, 9, 11, 13, 10} is not a *isSym* array.

From left to right we have {even, odd, even, odd, odd, odd, even}.

gramming in C or C++, the function signature is

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

ray is defined to be an array where for every value n in the array, there is also an element n-1 or n+1 in

, 2, 10, 9, 3} is a Nice array because

e arrays include {2, 2, 3, 3, 3}, {1, 1, 1, 2, 1, 1} and {0, -1, 1}.

, 4, 5, 7} is **not** a Nice array because of the value 7 which requires that the array contains either the value 6 or 8, but neither of these values are in the array.

ction named isNice that returns 1 if its array argument is a Nice array. Otherwise it returns a 0.

ogramming in Java or C#, the function signature is

ce int[] a)

ogramming in C or C++, the function signature is

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

There are 3 questions on this test. You have 2 hours to finish it. Please do your own work. All you need to write is three functions. Please do not use any string functions. No sorting allowed. No additional arrays allowed.

Write a function named **minDistance** that returns the smallest distance between two non-trivial factors of a number. For example, consider 63. Its non-trivialfactors are 3, 7, 9 and 21. Thus **minDistance**(63) would return 2 because the smallest distance between any two non-trivial factors is 2 (9 - 7 = 2). As another example, **minDistance**(25) would return 0 because 25 has only one non-trivial factor: 5. Thus the smallest distance between any two non-trivial factors is 0 (5 - 5 = 0). Note that **minDistance**(11) would return -1 since 11 has no non-trivial factors.

The function signature is

n **minDistance**(int n)

A **wave array** is defined to an array which does **not** contain two even numbers or two odd numbers in adjacent locations. So {7, 2, 9, 10, 5}, {4, ..., 12, ..., 6}, {1, 0, 5} and {2} are all **wave arrays**. But {**2**, **6**, 3, 4} is not a wave array because the even numbers 2 and 6 are adjacent to each other.

Write a function named isWave that returns 1 if its array argument is a Wave array, otherwise it returns 0.

{3, 3, 2, 6, 6, 7} because it has three even values.

{3, 3, 2, 7, 6, 7} because the even values are not adjacent to one another

{3, 8, 5, 7, 3} because it has only one even value.

Write a function named **isTwinoid** that returns 1 if its array argument is a twinoid array. Otherwise it returns 0.

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

int isTwinoid (int [ ] a);

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

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

3. A **balanced** array is defined to be an array where for every value n in the array, -n also is in the array. For example {-2, 3, 2, -3} is a balanced array. So is {-2, 2, 2, 2}. But {-5, 2, -2} is not because 5 is not in the array.

Write a function named isBalanced that returns 1 if its array argument is a balanced array. Otherwise it returns 0.

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

int isBalanced (int [ ] a);

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

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

18th of June 2016

rk. All you need to write is three functions. Please do not use any string methods. No sorting allowed. No additional data structures including arrays allowed. Try to write a simple, elegant and correct code.

1. **Mode** is the most frequently appearing value. Write a function named *hasSingleMode* that takes an array argument and returns 1 if the mode value in its array argument occurs exactly once in the array, otherwise it returns 0. If you are writing in Java or C#, the function signature is

int hasSingleMode(int[ ] a

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

int hasSingleMode(int a[ ], int len)

where len is the length of a.

Examples

Array elements	Mode values	Value returned	Comments
1, -29, 8, 5, -29, 6	-29	1	single mode
1, 2, 3, 4, 2, 4, 7	2, 4	0	no single mode
1, 2, 3, 4, 6	1, 2, 3, 4, 6	0	no single mode
7, 1, 2, 1, 7, 4, 2, 7	7	1	single mode