1.**Write a function that will iterate through an array *a* as follows.** Start at a[0]. If a[0] is -1 return -1. If a[0] is less than -1 or greater than or equal to the length of the array (i.e., it can't be used to index an element of the array), return 1. Otherwise visit a[a[0]] and repeat these steps. This could potentially result in an infinite loop. If an infinite loop is detected the function should return a 0.

To summarize:

1. iterate through the array using the value of an element as the index to the next element (like in a linked list)
2. return -1 if a -1 encountered
3. return 1 if a value less than -1 or greater than or equal to the size of the array is encountered.
4. return 0 if an infinite loop is detected.

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

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

Examples

if the input array is	Traversal	return
{1, 2, -1, 5}	visit a[0], a[1], a[2]	-1 (because -1 is encountered before the 5 is encountered)
{1, 2, 4, -1}	visit a[0], a[1], a[2]	1 (because 4, which is too big to be an index, is encountered before the -1)
{5, 3, 4, -1, 1, 2}	visit a[0], a[5], a[2], a[4], a[1], a[3]	-1 (because a[3] is -1)
{3}	visit a[0]	1 (because 3, which is too big to be an index, is encountered.)
{3, 2, 3, 1}	visit a[0], a[3], a[1], a[2], a[3], ...	0
{0}	visit a[0], a[0], ...	0
{-1}	visit a[0]	-1

Copy and paste your answer here and click the "Submit answer" button

Submit answer to question 1

You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

2. Define an array to be *cumulative* if the nth (n > 0) element of the array is the sum of the first n elements of the array. So {1, 1, 2, 4, 8} is cumulative because

1. a[1] == 1 == a[0]
2. a[2] == 2 == a[0] + a[1]
3. a[3] == 4 == a[0] + a[1] + a[2]
4. a[4] == 8 == a[0] + a[1] + a[2] + a[4]

And {1, 1, 2, 5, 9} is not cumulative because a[3] == 5 != a[0] + a[1] + a[2]

Write a function named isCumulative that accepts an array of integers and returns 1 if the array is cumulative and 0 otherwise.

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

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

Some other examples:

if the input array is	**isCumulative should return**
{1}	0 (array must contain at least 2 elements)
{0,0,0,0,0,0}	1
{1, 1, 1, 1, 1, 1}	0
{3, 3, 6, 12, 24}	1
{-3, -3, -6, -12, -24}	1
{-3, -3, 6, 12, 24}	0

Copy and paste your answer here and click the "Submit answer" button

Submit answer to question 2

You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

3. **Write a function that takes two arguments, an array of integers and a positive, non-zero number n. It sums n elements of the array starting at the beginning of the array.** If n is greater than the number of elements in the array, the function loops back to the beginning of the array and continues summing until it has summed n elements. You may assume that the array contains at least one element and that n is greater than 0.

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

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

Examples

If a is	and n is	then function returns
{1, 2, 3}	2	3 (which is a[0] + a[1])
{-1, 2, -1}	7	-1 (which is a[0] + a[1] + a[2] + a[0] + a[1] + a[2] + a[0])
{1, 4, 5, 6}	4	16 (which is a[0] + a[1] + a[2] + a[3])

Copy and paste your answer here and click the "Submit answer" button

Submit answer to question 3

You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

1 Write a function **int hasNFollowingComposites(int n, int count)** that returns 1 if *n* is a prime number and the next *count* numbers are composite(non-prime). Otherwise, it returns 0. Recall that a prime number is a number whose only factors are 1 and itself. You may assume that *n* and *count* are greater than zero.

Examples:

if n is	**and count is**	**Return**
23	5	1 because 23 is prime and the next 5 numbers, 24, 25, 26, 27 and 28 are composite
23	6	0 because 23 is prime but the 6th number following it (29) is prime, i.e., the next six numbers are not **all** composite.
89	6	1 because 89 is prime and the next 6 numbers, 90, 91(13*7), 92, 93(31*3), 94, 95 are composite.
24	4	0 because 24 is **not** prime (it doesn't matter that the next 4 numbers are composite)

Copy and paste your answer here and click the "Submit answer" button

Submit answer to question 1

You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

2 Write a function named **equivalentArrays** that has two array arguments and returns 1 if the two arrays contain the same values (but not necessarily in the same order), otherwise it returns 0. Your solution must **not** sort either array or a copy of either array! You may assume that both arrays have the same number of elements.

If you are programming in Java or C#, the function prototype is int equivalentArrays(int[ ] a1, int[ ] a2)

If you are programing in C or C++, the function prototype is int equivalentArrays(int a1[ ], int a2[ ], int len) where len is the number of elements in a1 and a2.

Examples:

if a1 is	and a2 is	Return
{0, 1, 2}	{2, 0, 1}	1
{1, 1, 1, 1, 1, 1}	{1, 1, 1, 1, 1, 2}	0 because every element of a1 is in a2 but not vice versa.
{}	{3, 1, 1, 1, 1, 2}	0 because every element of a1 (show me one that isn't) is in a2 but not vice versa.

Copy and paste your answer here and click the "Submit answer" button

Submit answer to question 2

You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

3 Write a function named **hasTwoValues** which takes an array as an argument. It returns 1 if all the elements of the array are one of two different values, otherwise it returns 0. Your solution must make exactly one pass through the array. It must not sort the array or a copy of the array.. Furthermore, your solution must not have any nested loops in it! We are looking for a solution that minimizes time. Note that an element of the array can be any value including 0, negative numbers, the maximum integer and the minimum integer.

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

If you are writing in C or C++, the function signature is
int hasTwoValues(int a[ ], int len) where len is the length of a

Examples

if a is	Return
{1, 2, 2, 1}	1 (because there are 2 different element values)
{1, 1, 1, 8, 1, 1, 1, 3, 3}	0 (because there are 3 different element values, 1, 3, 8, not two as required.)
{1, 1, 1, 1, 1, 1, 1, 1, 1}	0 (because there is only one element value, 1, not two as required.)

{}	0 (because there are 0 different element values, not two as required.)

Hint: One possible solution uses the following local variables

```
  int cnt1 = 0, cnt2 = 0;
  int n1=0, n2=0;
```

Copy and paste your answer here and click the "Submit answer" button

Submit answer to question 3

You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

1. Write a function named *isSorted* that accepts an integer array and returns 1 if its elements are in ascending or descending order, otherwise it returns 0.

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

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

Examples:

if the input array is	**Return**
{1, 2, 5, 6}	1
{12, 3, 2, 1}	1
{1, 2, 6, 3}	0 (because it is in neither ascending or descending order)
{}	1
{2}	1

Copy and paste your answer here and click the "Submit answer" button

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

if the number is	Return
0	1
1	1
2	1
3	1
4	1
5	1
6	0 (3 is a factor)
7	1
8	1
9	0 (3 is a factor)
10	0 (5 is a factor)
11	1
12	0 (3 is a factor)
13	1
14	0 (7 is a factor)
15	0 (3 and 5 are factors)
16	1
17	1
18	0 (3 is a factor)
19	1
20	0 (5 is a factor)

Copy and paste your answer here and click the "Submit answer" button

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

Copy and paste your answer here and click the "Submit answer" button

Submit answer to question 1

You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

2. **A non-empty array *a* of length n is called an array of all possiblities** if it contains all numbers between 0 and a.length-1 inclusive. Write a method named **isAllPossibilities** that accepts an integer array and returns 1 if the array is an array of all possiblities, otherwise it returns 0.

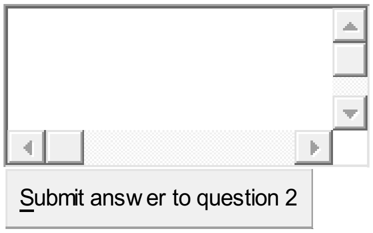
If you are programming in Java or C#, the function signature is
int isAllPossibilities(int[ ] a)

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

Examples

if the input array is	Return
{1, 2, 0, 3}	1
{3, 2, 1, 0}	1
{1, 2, 4, 3}	0 (because 0 not included and 4 is too big)
{0, 2, 3}	0 (because 1 is not included)
{0}	1
{}	0

Copy and paste your answer here and click the "Submit answer" button



You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

3. **An array is called *layered*** if its elements are in ascending order and each element appears two or more times. For example, {1, 1, 2, 2, 2, 3, 3} is layered but {1, 2, 2, 2, 3, 3} and {3, 3, 1, 1, 1, 2, 2} are not. Write a method named **isLayered** that accepts an integer array and returns 1 if the array is layered, otherwise it returns 0.

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

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

Examples:

if the input array is	Return
{1, 1, 2, 2, 2, 3, 3}	1
{3, 3, 3, 3, 3, 3, 3}	1
{1, 2, 2, 2, 3, 3}	0 (because there is only one occurence of the value 1)

{2, 2, 2, 3, 3, 1, 1}	0 (because values are not in ascending order)
{2}	0
{}	0

Copy and paste your answer here and click the "Submit answer" button

Submit answer to question 3

You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

1. Write a function that accepts an array of non-negative integers and returns the second largest integer in the array. Return -1 if there is no second largest.
The signature of the function is
int f(int[ ] a)

Examples:

if the input array is	Return
{1, 2, 3, 4}	3
{{4, 1, 2, 3}}	3
{1, 1, 2, 2}	1
{1, 1}	-1
{1}	-1
{}	-1

Copy and paste your answer here and click the "Submit answer" button

In the real exam you would pas

Answer to first question

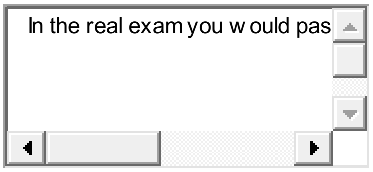
2. Write a function that takes an array of integers as an argument and returns a value based on the sums of the even and odd numbers in the array. Let X = the sum of the odd numbers in the array and let Y = the sum of the even numbers. The function should return X - Y

The signature of the function is:
int f(int[ ] a)

Examples

if input array is	return
{1}	1
{1, 2}	-1
{1, 2, 3}	2
{1, 2, 3, 4}	-2
{3, 3, 4, 4}	-2
{3, 2, 3, 4}	0
{4, 1, 2, 3}	-2
{1, 1}	2
{}	0

Copy and paste your answer here and click the "Submit answer" button



In the real exam you w ould pas

Answer to second question

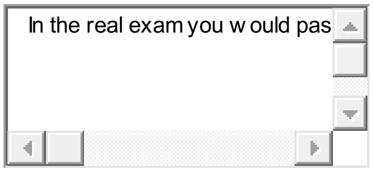
3. Write a function that accepts a character array, a zero-based start position and a length. It should return a character array containing containing *length* characters starting with the *start* character of the input array. The function should do error checking on the start position and the length and return null if the either value is not legal. The function signature is:
char[ ] f(char[ ] a, int start, int len)

Examples

if input parameters are	Return
{'a', 'b', 'c'}, 0, 4	Null

{'a', 'b', 'c'}, 0, 3	{'a', 'b', 'c'}
{'a', 'b', 'c'}, 0, 2	{'a', 'b'}
{'a', 'b', 'c'}, 0, 1	{'a'}
{'a', 'b', 'c'}, 1, 3	Null
{'a', 'b', 'c'}, 1, 2	{'b', 'c'}
{'a', 'b', 'c'}, 1, 1	{'b'}
{'a', 'b', 'c'}, 2, 2	Null
{'a', 'b', 'c'}, 2, 1	{'c'}
{'a', 'b', 'c'}, 3, 1	Null
{'a', 'b', 'c'}, 1, 0	{}
{'a', 'b', 'c'}, -1, 2	Null
{'a', 'b', 'c'}, -1, -2	Null
{}, 0, 1	Null

Copy and paste your answer here and click the "Submit answer" button



In the real exam you would pas

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.

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

Examples:

if the input array is	**return**	**Reason**
{2, 3, 6, 7}	1	a[0] and a[2] are even, a[1] and a[3] are odd.
{6, 3, 2, 7}	1	a[0] and a[2] are even, a[1] and a[3] are odd.
{6, 7, 2, 3, 12}	1	a[0], a[2] and a[4] are even, a[1] and a[3] are odd.
{6, 7, 2, 3, 14, 95}	1	a[0], a[2], and a[4] are even, a[1], a[3] and a[5] are odd.

{7, 15, 2, 3}	0	a[0] is odd
{16, 6, 2, 3}	0	a[1] is even
{2}	1	a[0] is even
{3}	0	a[0] is odd
{}	1	true vacuously

Copy and paste your answer here and click the "Submit answer" button

Submit answer to question 1

You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

2. Write a function named *eval* that returns the value of the polynomial $a_nx^n + a_{n-1}x^{n-1} + ... + a_1x^1 + a_0$.

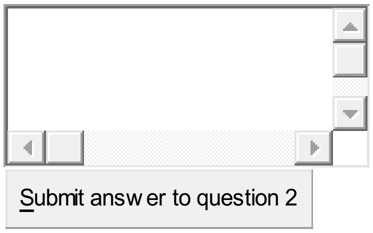
If you are programming in Java or C#, the function signature is
double eval(double x, int[ ] a)

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

Examples:

if x is	**if the input array is**	**this represents**	**eval should return**
1.0	{0, 1, 2, 3, 4}	$4x^4 + 3x^3 + 2x^2 + x + 0$	10.0
3.0	{3, 2, 1}	$x^2 + 2x + 3$	18.0
2.0	{3, -2, -1}	$-x^2 - 2x + 3$	-5.0
-3.0	{3, 2, 1}	$x^2 + 2x + 3$	6.0
2.0	{3, 2}	$2x + 3$	7.0
2.0	{4, 0, 9}	$9x^2 + 4$	40.0
2.0	{10}	10	10.0
10.0	{0, 1}	X	10.0

Copy and paste your answer here and click the "Submit answer" button



You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

3. A palindrome is a word or phase that reads the same backwards or forwards. Write a function named *isPalindrome* that returns true or false if the input array is a palindrome.

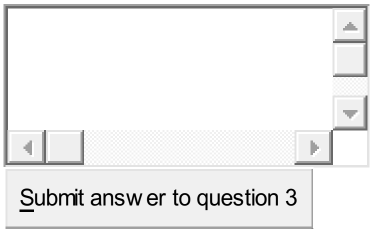
If you are programming in Java or C#, the function signature is
boolean isPalindrome(char [] arr)

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

Examples:

if the input array is	**output is**
{'t', 'o', 'p', 's', 'p', 'o', 't'}	True
{'t','o','t','o'}	False
{'d','o','t','s','e','e','s','t','o','d'}	True
{ }	False
{'a'}	True
{4, 0, 9}	False
the char string "ipreferpi"	True
{0, 1, 0}	True

Copy and paste your answer here and click the "Submit answer" button



Submit answer to question 3

You should see a confirmation popup after hitting the submit button above. If you do not see a confirmation popup, please email your answer.

1. Write a function named *isMaxnumberelementfourtime* that returns *Maxnumberelementfourtime*.

int arr(int[ ] a)
Ex: {11,11,2,3,4,5,5,6,7,5,5,7,11,7,7}

2. Write a function named *Thesamdegit* that returns *1 is all digit the same other return 0*. If overflow return -1

int ***Thesamdegit*** (int n)
Ex: 8887