



BUY OPENCART MODULES TO ENHANCE SITE
DpSignAdvertising.com OR OpencartNepal.com

[OPENCART](#)[ECOMMERCE](#)[WORDPRESS](#)[PHP](#)[INSPIRATION](#)[ENTERTAINMENT](#)[TOOLS](#)[OTHER](#)[Commerce Store](#)You Are Here: [Home](#) » [Other](#) » Sample 1 questions answer for the MUM entrance exam

Sample 1 questions answer for the MUM entrance exam

Posted by: [rupaknpl](#) Posted date: **February 16, 2013** In: [Other](#) | comment : **3**

After applying in the Maharishi University of Management (MUM) for Master of Science in Computer Science, we get student id number provided by the University and we have to appear to the entrance exam, and in the entrance exam you will have three questions to be solved within 2 hours. I have collected some of the question and try to provide you the answers. Some of the sample questions that i am able to collect for the MUM entrance exam are as follows:

There are three questions on the exam. You have two hours to finish. **Please do your own work.**

1. Write a function named **primeCount** with signature

```
int primeCount(int start, int end);
```

The function returns the number of primes between *start* and *end* inclusive. Recall that a prime is a positive integer greater than 1 whose only integer factors are 1 and itself.

Examples

If start is	and end is	return	reason
10	30	6	The primes between 10 and 30 inclusive are 11, 13, 17, 19, 23 and 29
11	29	6	The primes between 11 and 29 inclusive are 11, 13, 17, 19, 23 and 29
20	22	0	20, 21, and 22 are all non-prime
1	1	0	By definition, 1 is not a prime number
5	5	1	5 is a prime number
6	2	0	start must be less than or equal to end
-10	6	3	primes are greater than 1 and 2, 3, 5 are prime

2. A **Madhav** array has the following property.

$$a[0] = a[1] + a[2] = a[3] + a[4] + a[5] = a[6] + a[7] + a[8] + a[9] = \dots$$

The length of a Madhav array must be $n*(n+1)/2$ for some n .

Write a method named *isMadhavArray* that returns 1 if its array argument is a Madhav array, otherwise it returns 0. If you are programming in Java or C# the function signature is

```
int isMadhavArray(int[] a)
```

Rupak Nepali - PHP Programmer and Opencart Proficient, Nepal



\$30/hr ★★★★★ (4.5)

428 oDesk Hours | 36

Contracts

Agency: Self Agency > DP

Sign Pvt...Self Agency > DP Sign Pvt...

- Three years of experience as PHP programmer
- Excellent team player with positive attitude
- Strong presentation and... [more](#)

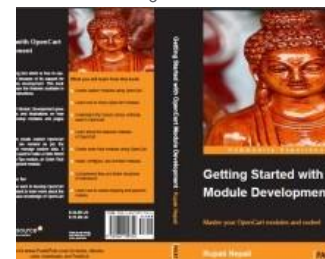
Hire me on **oDesk**

Want to help me below are the links :)

Donate



Like to excel in OpenCart then you must have to read the following book



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

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

Examples

if a is	return	reason
{2, 1, 1}	1	$2 + 1 + 1$
{2, 1, 1, 4, -1, -1}	1	$2 = 1 + 1, 2 = 4 + -1 + -1$
{6, 2, 4, 2, 2, 2, 1, 5, 0, 0}	1	$6 = 2 + 4, 6 = 2 + 2 + 2, 6 = 1 + 5 + 0 + 0$
{18, 9, 10, 6, 6, 6}	0	$18 \neq 9 + 10$
{-6, -3, -3, 8, -5, -4}	0	$-6 \neq 8 + -5 + -4$
{0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, -2, -1}	1	$0 = 0 + 0, 0 = 0 + 0 + 0, 0 = 0 + 0 + 0 + 0,$ $0 = 1 + 1 + 1 + -2 + -1$
{3, 1, 2, 3, 0}	0	The length of the array is 5, but 5 does not equal $n*(n+1)/2$ for any value of n.

3. An array is defined to be **inertial** if the following conditions hold:

- a. it contains at least one odd value
- b. the maximum value in the array is even
- c. every odd value is greater than every even value that is not the maximum value.

So {11, 4, 20, 9, 2, 8} is inertial because

- a. it contains at least one odd value
- b. the maximum value in the array is 20 which is even
- c. the two odd values (11 and 9) are greater than all the

even values that are not equal to 20 (the maximum), i.e., (4, 2, 8).

However, {12, 11, 4, 9, 2, 3, 10} is **not** inertial because it fails condition (c), i.e., 10 (which is even) is greater 9 (which is odd) but 10 is not the maximum value in the array.

Write a function called *isInertial* that accepts an integer array and returns 1 if the array is inertial; otherwise it returns 0.

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

int isInertial(int[] a)

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

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

Some other examples:

if the input array is	return	reason
{1}	0	fails condition (a) – the maximum value must be even
{2}	0	fails condition (b) – the array must contain at least one odd value.



UpwardlyGlobal
Specialized job search resources for
global tech and engineering
professionals

*Restart Your Career
in the United States*

+1 at Google Plus

WebOCreation.com

google.com/+RupaknepaliNp

e-commerce, e-commerce application developer,
e-commerce programmer

READ PREVIOUS POST:



[RestaurantKathmandu.lk with the new backend system and design and one time popup module](#)

RestaurantKathmandu.lk with the new backend system and design. Let start to become the member and never miss the news, its...



UpwardlyGlobal
Specialized job search resources for
global tech and engineering
professionals

*Restart Your Career
in the United States*

{1, 2, 3, 4}	0	fails condition (c) – 1 (which is odd) is not greater than all even values other than the maximum (1 is less than 2 which is not the maximum)
{1, 1, 1, 1, 1, 2}	1	there is no even number other than the maximum. Hence, there can be no other even values that are greater than 1.
{2, 12, 4, 6, 8, 11}	1	11, the only odd value is greater than all even values except 12 which is the maximum value in the array.
{2, 12, 12, 4, 6, 8, 11}	1	same as previous, i.e., it is OK if maximum value occurs more than once.
{-2, -4, -6, -8, -11}	0	-8, which is even, is not the maximum value but is greater than -11 which is odd
{2, 3, 5, 7}	0	the maximum value is odd
{2, 4, 6, 8, 10}	0	there is no odd value in the array.

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

1. Define a **square pair** to be the tuple $\langle x, y \rangle$ where x and y are positive, non-zero integers, $x < y$ and $x + y$ is a perfect square. A perfect square is an integer whose square root is also an integer, e.g. 4, 9, 16 are perfect squares but 3, 10 and 17 are not. Write a function named *countSquarePairs* that takes an array and returns the number of square pairs that can be constructed from the elements in the array. For example, if the array is {11, 5, 4, 20} the function would return 3 because the only square pairs that can be constructed from those numbers are $\langle 5, 11 \rangle$,

$\langle 5, 20 \rangle$ and $\langle 4, 5 \rangle$. **You may assume that there exists a function named *isPerfectSquare* that returns 1 if its argument is a perfect square and 0 otherwise. E.G., *isPerfectSquare(4)* returns 1 and *isPerfectSquare(8)* returns 0.**

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

```
int countSquarePairs(int[] a)
```

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

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

You may assume that there are no duplicate values in the array, i.e, you don't have to deal with an array like {2, 7, 2, 2}.

Examples:

if a is	return	reason
{9, 0, 2, -5, 7}	2	The square pairs are $\langle 2, 7 \rangle$ and $\langle 7, 9 \rangle$. Note that $\langle -5, 9 \rangle$ and $\langle 0, 9 \rangle$ are not square pairs, even though they sum to perfect squares, because both members of a square pair have to be greater than 0. Also $\langle 7, 2 \rangle$ and $\langle 9, 7 \rangle$ are not square pairs because the first number has to be less than the second number.
{9}	0	The array must have at least 2 elements

2. A **prime number** is an integer that is divisible only by 1 and itself. A **porcupine number** is a prime number whose last digit is 9 and the next prime number that follows it also ends with the digit 9. For example 139 is a porcupine number because:

a. it is prime

b. it ends in a 9

c. The next prime number after it is 149 which also ends in 9. Note that 140, 141, 142, 143, 144, 145, 146, 147 and 148 are **not** prime so 149 is the next prime number after 139.

Write a method named *findPorcupineNumber* which takes an integer argument *n* and returns the first porcupine number **that is greater than *n***. So *findPorcupineNumber*(0) would return 139 (because 139 happens to be the first porcupine number) and so would *findPorcupineNumber*(138). But *findPorcupineNumber*(139) would return 409 which is the second porcupine number.

The function signature is

```
int findPorcupineNumber(int n)
```

You may assume that a porcupine number greater than *n* exists.

You may assume that a function *isPrime* exists that returns 1 if its argument is prime, otherwise it returns 0. E.G., *isPrime*(7) returns 1 and *isPrime*(8) returns 0.

Hint: Use modulo base 10 arithmetic to get last digit of a number.

3. Consider the following algorithm

*Start with a positive number *n**

*if *n* is even then divide by 2*

*if *n* is odd then multiply by 3 and add 1*

*continue this until *n* becomes 1*

The **Guthrie sequence** of a positive number *n* is defined to be the numbers generated by the above algorithm.

For example, the Guthrie sequence of the number 7 is

7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

It is easy to see that this sequence was generated from the number 7 by the above algorithm. Since 7 is odd multiply by 3 and add 1 to get 22 which is the second number of the sequence. Since 22 is even, divide by 2 to get 11 which is the third number of the sequence. 11 is odd so multiply by 3 and add 1 to get 34 which is the fourth number of the sequence and so on.

Note: the first number of a Guthrie sequence is always the number that generated the sequence and the last number is always 1.

Write a function named *isGuthrieSequence* which returns 1 if the elements of the array form a Guthrie sequence. Otherwise, it returns 0.

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

```
int isGuthrieSequence(int[] a)
```

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

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

Examples

if a is	return	reason
{8, 4, 2, 1}	1	This is the Guthrie sequence for 8
{8, 17, 4, 1}	0	This is not the Guthrie sequence for 8
{8, 4, 1}	0	Missing the 2
{8, 4, 2}	0	A Guthrie sequence must end with 1

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

1. The **Stanton measure** of an array is computed as follows. Count the number of 1s in the array. Let this count be n . The Stanton measure is the number of times that n appears in the array. For example, the Stanton measure of {1, 4, 3, 2, 1, 2, 3, 2} is 3 because 1 occurs 2 times in the array and 2 occurs 3 times.

Write a function named *stantonMeasure* that returns the Stanton measure of its array argument.

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

```
int stantonMeasure(int[] a)
```

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

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

Examples

if a is	return	reason
{1}	1	1 occurs 1 time, 1 occurs 1 time
{0}	1	1 occurs 0 times, 0 occurs 1 time
{3, 1, 1, 4}	0	1 occurs 2 times, 2 occurs 0 times
{1, 3, 1, 1, 3, 3, 2, 3, 3, 4}	6	1 occurs 3 times, 3 occurs 6 times
{}	0	1 occurs 0 times, 0 occurs 0 times

2. The **sum factor** of an array is defined to be the number of times that the sum of the array appears as an element of the array. So the sum factor of {1, -1, 1, -1, 1, -1, 1} is 4 because the sum of the elements of the array is 1 and 1 appears four times in the array. And the sum factor of

{1, 2, 3, 4} is 0 because the sum of the elements of the array is 10 and 10 does not occur as an element of the array. The sum factor of the empty array {} is defined to be 0.

Write a function named *sumFactor* that returns the sum factor of its array argument.

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

```
int sumFactor(int[] a)
```

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

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

Examples:

if a is	return	reason
{3, 0, 2, -5, 0}	2	The sum of array is 0 and 0 occurs 2 times
{9, -3, -3, -1, -1}	0	The sum of the array is 1 and 1 does not occur in array.
{1}	1	The sum of the array is 1 and 1 occurs once in the array
{0, 0, 0}	3	The sum of the array is 0 and 0 occurs 3 times in the array

3. Consider the following algorithm

Start with a positive number n

if n is even then divide by 2

if n is odd then multiply by 3 and add 1

continue this until n becomes 1

The **Guthrie index** of a positive number n is defined to be how many iterations of the above algorithm it takes before n becomes 1.

For example, the Guthrie index of the number 7 is 16 because the following sequence is 16 numbers long.

22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

It is easy to see that this sequence was generated by the above algorithm. Since 7 is odd multiply by 3 and add 1 to get 22 which is the first number of the sequence. Since 22 is even, divide by 2 to get 11 which is the second number of the sequence. 11 is odd so multiply by 3 and add 1 to get 34 which is the third number of the sequence and so on.

Write a function named *guthrieIndex* which computes the Guthrie index of its argument. Its signature is

int guthrieIndex(int n)

Examples

if n is	return	sequence
1	0	number is already 1
2	1	1
3	7	10, 5, 16, 8, 4, 2, 1
4	2	2, 1
42	8	21, 64, 32, 16, 8, 4, 2, 1

You may assume that the length of the sequence can be represented by a 32 bit signed integer.

There are 3 questions on this test. You have two hours to do it. **Please do your own work.**

1. It is a fact that there exist two numbers x and y such that $x! + y! = 10!$. Write a method named **solve10** that returns the values x and y in an array.

The notation $n!$ is called **n factorial** and is equal to $n * n-1 * n-2 * \dots * 2 * 1$, e.g., $5! = 5*4*3*2*1 = 120$.

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

int[] solve10() where the length of the returned array is 2.

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

int* solve10() where the length of the returned array is 2.

Please be sure that the method solve10 returns an array, a , with two elements

where $a[0] == x$, $a[1] == y$ and $x! + y! = 10!$.

2. An array can hold the digits of a number. For example the digits of the number 32053 are stored in the array {3, 2, 0, 5, 3}. Write a method call **repsEqual** that takes an array and an integer and returns 1 if the array contains **only** the digits of the number **in the same order** that they appear in the number. Otherwise it returns 0.

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

int repsEqual(int[] a, int n)

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

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

Examples (note: your program must work for all values of a and n , not just those given here!)

if a is	and n is	return	reason
{3, 2, 0, 5, 3}	32053	1	the array contains only the digits of the number, in the same order as they are in the number.
{3, 2, 0, 5}	32053	0	the last digit of the number is missing from the array.
{3, 2, 0, 5, 3, 4}	32053	0	an extra number (4) is in the array.
{2, 3, 0, 5, 3}	32053	0	the array elements are not in the same order as the digits of the number
{9, 3, 1, 1, 2}	32053	0	elements in array are not equal to digits of number.
{0, 3, 2, 0, 5, 3}	32053	1	you can ignore leading zeroes.

3. An array is called **centered-15** if some consecutive sequence of elements of the array sum to 15 and this sequence is preceded and followed by the same number of elements. For example

{3, 2, 10, 4, 1, 6, 9} is centered-15 because the sequence 10, 4, 1 sums to 15 and the sequence is preceded by two elements (3, 2) and followed by two elements (6, 9).

Write a method called `isCentered15` that returns 1 if its array argument is centered-15, otherwise it returns 0.

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

```
int isCentered15(int[] a)
```

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

```
int isCentered5(int a[], int len) where len is the number of elements in the array.
```

Examples

if a is	return	reason
{3, 2, 10, 4, 1, 6, 9}	1	the sequence 10, 4, 1 sums to 15 and is preceded by 2 elements and followed by 2 elements. Note that there is another sequence that sums to 15 (6, 9). It is okay for the array to have more than one sequence that sums to 15 as long as at least one of them is centered.
{2, 10, 4, 1, 6, 9}	0	(10, 4, 1) is preceded by one element but followed by two. (9, 6) is preceded by five elements but followed by none. Hence neither qualify as centered.
{3, 2, 10, 4, 1, 6}		(10, 4, 1) is preceded by two elements but followed by one. Note that the values 3, 2, 4, 6 sum to 15 but they are not consecutive.
{1, 1, 8, 3, 1, 1}		The entire array sums to 15, hence the sequence is preceded by zero elements and followed by zero elements.
{9, 15, 6}	1	the sequence (15) is preceded by one element and followed by one element.
{1, 1, 2, 2, 1, 1}	0	no sequence sums to 15.
{1, 1, 15, -1, -1}	1	there are three different sequences that sum to 15, the entire array, (1, 15, -1) and (15). In this case they all are centered but the requirement is that just one of them has to be.

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

1. A **perfect number** is one that is the sum of its factors, excluding itself. The 1st perfect number is 6 because $6 = 1 + 2 + 3$. The 2nd perfect number is 28 which equals $1 + 2 + 4 + 7 + 14$. The third is $496 = 1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248$. In each case, the number is the sum of all its factors excluding itself.

Write a method named *henry* that takes two integer arguments, *i* and *j* and returns the sum of the *i*th and *j*th perfect numbers. So for example, *henry* (1, 3) should return 502 because 6 is the 1st perfect number and 496 is the 3rd perfect number and $6 + 496 = 502$.

The function signature is

```
int henry (int i, int j)
```

You do not have to worry about integer overflow, i.e., you may assume that each sum that you have to compute can be represented as a 31 bit integer. Hint: use modulo arithmetic to determine if one number is a factor of another.

2. Write a method named *isDivisible* that takes an integer array and a divisor and returns 1 if all its elements are divided by the divisor with no remainder. Otherwise it returns 0.

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

```
int isDivisible(int [ ] a, int divisor)
```

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

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

Examples

if a is	and divisor is	return	because
{3, 3, 6, 36}	3	1	all elements of a are divisible by 3 with no remainder.
{4}	2	1	all elements of a are even
{3, 4, 3, 6, 36}	3	0	because when a[1] is divided by 3, it leaves a remainder of 1
{6, 12, 24, 36}	12	0	because when a[0] is divided by 12, it leaves a remainder of 6.
{}	anything	1	because no element fails the division test.

3. An array is defined to be **n-unique** if exactly one pair of its elements sum to *n*. For example, the array {2, 7, 3, 4} is 5-unique because only a[0] and a[2] sum to 5. But the array {2, 3, 3, 7} is not 5-unique because a[0] + a[1] = 5 and a[0] + a[2] = 5.

Write a function named *isNUnique* that returns 1 if its integer array argument is n-unique, otherwise it returns 0. So *isNUnique*(new int[] {2, 7, 3, 4}, 5) should return 1 and

isNUnique(new int[] {2, 3, 3, 7}, 5) should return 0.

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

```
int isNUnique(int[] a, int n)
```

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

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

Examples

If a is	and n is	return	because
---------	----------	--------	---------

{7, 3, 3, 2, 4}	6	0	because $a[1]+a[2] == 6$ and $a[3]+a[4]$ also $== 6$.
{7, 3, 3, 2, 4}	10	0	because $a[0]+a[1] == 10$ and $a[0] + a[2]$ also $== 10$
{7, 3, 3, 2, 4}	11	1	because only $a[0] + a[4]$ sums to 11
{7, 3, 3, 2, 4}	8	0	because no pair of elements sum to 8. (Note that $a[1]+a[2]+a[3]$ do sum to 8 but the requirement is that two elements sum to 8.)
{7, 3, 3, 2, 4}	4	0	no pair of elements sum to 4. (Note that the $a[4]==4$, but the requirement is that two elements have to sum to 4.)
{1}	anything	0	array must have at least 2 elements

0

0

0

0

Tweet

+1

Like

Share


Share

-
-


< Previous

Next >


Related posts



Scubattokki.com checkout system and many vqmod conflicts removed successfully




2000th title of Packt Publishing are launching an exciting campaign until 26th Mar 2014



How Reward Points works in opencart 1.5+ eCommerce site

MUM test entrance exam solutions

Comments (3)




mother fucker

July 31 at

where is the solution u crazy fuck!!!!

Reply




Tan

September 24 at

Thanks for the effort bro

Reply



Ramesh

December 24 at

We would be glad if we got answer of those questions

Reply

Leave a Comment

<input type="text"/>	Name*
<input type="text"/>	Email*
<input type="text"/>	Website
<div></div>	
<input type="button" value="Submit Comment"/>	
<input type="checkbox"/>	Notify me of follow-up comments by email.
<input type="checkbox"/>	Notify me of new posts by email.