**Computer Professionals**

**Master’s Program Sample Test**

You will be asked to complete a real test as part of your application process.

The purpose of this short test is to assess your ability to solve elementary programming problems in a language of your choice.

Write your solutions in Java if you are familiar with that language; otherwise use one of these languages: C, C++, or C#. For each of the problems below, write the simplest, clearest solution you can, in the form of a short program.

**SAMPLE TEST**

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 that accepts an integer array and returns 1 if it is a centered array, otherwise it returns 0.**

**Examples:**

|  |  |
| --- | --- |
| **if the input array is** | **return** |
| {1, 2, 3, 4, 5} | 0 (the middle element 3 is not strictly less than all other elements) |
| {3, 2, 1, 4, 5} | 1 (the middle element 1 is strictly less than all other elements) |
| {3, 2, 1, 4, 1} | 0 (the middle element 1 is not strictly less than all other elements) |
| {1, 2, 3, 4} | 0 (no middle element) |
| {} | 0 (no middle element) |
| {10} | 1 (the middle element 10 is strictly less than all other elements) |

**Answers**

**First answer**

static int a1(int[] a)

{

if (a == null || a.length % 2 == 0) return 0;

int midIndex = a.length / 2 ;

int middleItem = a[midIndex];

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

{

if (i != midIndex && middleItem >= a[i])

return 0;

}

return 1;

}

1. **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 |

**Second answer**

static int a2(int[] a)

{

int sumEven = 0;

int sumOdd = 0;

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

{

if (a[i]%2 == 0)

sumEven += a[i];

else

sumOdd += a[i];

}

return sumOdd - sumEven;

}

1. **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 |

**Third answer**

static char[] a3(char[] a, int start, int length)

{

if (length < 0 || start < 0 || start+length-1>=a.length)

{

return null;

}

char[] sub = new char[length];

for (int i=start, j=0; j<length; i++, j++)

{

sub[j] = a[i];

}

return sub;

}

1. **Write a function to reverse an integer using numeric operators and without using any arrays or other data structures.**  
   The signature of the function is:  
   **int f(int n)**

**Examples**

|  |  |
| --- | --- |
| **if the input integer is** | **return** |
| 1234 | 4321 |
| 12005 | 50021 |
| 1 | 1 |
| 1000 | 1 |
| 0 | 0 |
| -12345 | -54321 |

1. **Write a function to return an array containing all elements common to two given arrays containing distinct positive integers. You should not use any inbuilt methods. You are allowed to use any number of arrays.**  
   The signature of the function is:  
   **int[] f(int[] first, int[] second)**

**Examples**

|  |  |
| --- | --- |
| **if input parameters are** | **return** |
| {1, 8, 3, 2}, {4, 2, 6, 1} | {1, 2} |
| {1, 8, 3, 2, 6}, {2, 6, 1} | {2, 6, 1} |
| {1, 3, 7, 9}, {7, 1, 9, 3} | {1, 3, 7, 9} |
| {1, 2}, {3, 4} | {} |
| {}, {1, 2, 3} | {} |
| {1, 2}, {} | {} |
| {1, 2}, null | null |
| null, {} | null |
| null, null | null |

**Fourth answer**

static int a4(int n)

{

int sign = 1;

if (n == 0) return 0;

if (n < 0)

{

sign = -1;

n = -n;

}

int reverse = 0;

while (n != 0)

{

reverse = (reverse \* 10) + (n % 10);

n /= 10;

}

return sign \* reverse;

}

1. **Write a function to return an array containing all elements common to two given arrays containing distinct positive integers. You should not use any inbuilt methods. You are allowed to use any number of arrays.**  
   The signature of the function is:  
   **int[] f(int[] first, int[] second)**

**Examples**

|  |  |
| --- | --- |
| **if input parameters are** | **return** |
| {1, 8, 3, 2}, {4, 2, 6, 1} | {1, 2} |
| {1, 8, 3, 2, 6}, {2, 6, 1} | {2, 6, 1} |
| {1, 3, 7, 9}, {7, 1, 9, 3} | {1, 3, 7, 9} |
| {1, 2}, {3, 4} | {} |
| {}, {1, 2, 3} | {} |
| {1, 2}, {} | {} |
| {1, 2}, null | null |
| null, {} | null |
| null, null | null |

**Fifth answer**

static int[] a5(int[] first, int[] second)

{

if (first == null || second == null) return null;

if (first.length == 0 || second.length == 0) return new int[0];

int min =

(first.length < second.length) ? first.length : second.length;

int[] a, b;

if (min == first.length)

{

a = first;

b = second;

}

else

{

a = second;

b = first;

}

int[] c = new int[min];

int k = 0;

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

for (int j = 0; j < b.length; j++)

if (a[i] == b[j])

{

c[k] = a[i];

k++;

}

int[] retArray = new int[k];

for (int t = 0; t < retArray.length; t++)

retArray[t] = c[t];

return retArray;

}

1. **Consider an array A with n of positive integers. An integer idx is called a POE (point of equilibrium) of A, if A[0] + A[1] + … + A[idx – 1] is equal to A[idx + 1] + A[idx + 2] + … + A[n – 1]. Write a function to return POE of an array, if it exists and -1 otherwise.**  
   The signature of the function is:  
   **int f(int[] a)**

**Examples**

|  |  |
| --- | --- |
| **if input arrays are** | **return** |
| {1, 8, 3, 7, 10, 2} | 3 Reason: a[0] + a[1] + a[2] is equal to a[4] + a[5] |
| {1, 5, 3, 1, 1, 1, 1, 1, 1} | 2 Reason: a[0] + a[1] is equal to a[3] + a[4] + a[5] + a[6] + a[7] + a[8] |
| {2, 1, 1, 1, 2, 1, 7} | 5 Reason: a[0] + a[1] + a[2] + a[3] + a[4] is equal to a[6] |
| {1, 2, 3} | -1 Reason: No POE. |
| {3, 4, 5, 10} | -1 Reason: No POE. |
| {1, 2, 10, 3, 4} | -1 Reason: No POE. |

**Sixth answer**

static int a6(int[] a)

{

if (a.length < 3) return -1;

int i = 0;

int j = a.length - 1;

int idx = 1;

int leftSum = a[i];

int rightSum = a[j];

for (int k = 1; k < a.length - 2; k++)

{

if (leftSum < rightSum)

{

i++;

leftSum += a[i];

idx = i + 1;

}

else

{

j--;

rightSum += a[j];

idx = j - 1;

}

}

if (leftSum == rightSum)

return idx;

else

return -1;

}

### Common Programming Exam Errors

Here is a list of common errors that students have committed in previous exams.  
You will receive no credit for an answer that commits any of these errors!

1. You are not allowed to collaborate with another person before, during or after the exam. The work you submit must be your own. If we detect an answer that was clearly shared amongst two or more applicants, we will stop grading the test and all applicants involved will fail the test!! Don’t collaborate!!
2. Do not hard-code the length of an array. Just because the examples require an array of length 3, say, does not mean that your program should be able to only handle arrays of length 3. Your program should be able to handle an array of any size! So if we see something like  
   **int[ ] a = new int[3];**  
   we will stop grading and give you zero points for the answer.
3. Do not use any I/O in your functions! You should not read in anything from the keyboard or write out anything to the console. A function should get its input from its parameters and return its answer using a return statement. Specifically, your function must be able to be used as is in a GUI-based windows application or in a browser-based web application. So if we see any of the following statements in your function, we will stop grading and you will receive no credit for your answer.
   1. cout
   2. cin
   3. printf
   4. scanf
   5. Console.WriteLine();
   6. Console.ReadLine();
   7. System.out.println();
   8. System.in.read();
   9. Any other I/O statement

It is okay to use output statements in your main program to display the results of calls to your function. But it is a waste of time to use input statements in your main program since it is okay to hard-code the values of any arrays that you use to test your function.