

**Developer Test**

**Applicant Name:** Ferenc Hartmann

**Date:** 2017.07.24.

**The following test serves to measure your technical knowledge.**

**Do not utilize any external resource during the test. It is better to hand in a partial answer on your own than relying on external help just to submit full solutions.**

**Write down your thoughts whenever necessary for our better understanding. If you are running out of time, please, try to share your thought about how would you write the code if time had allowed.**

**Please use English in your answers.**

**You have 100 minutes to complete this test.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Part I: Theory**

1. **What does *method overloading* mean?**

Method Overloading is a feature that allows a class to have two or more methods having same name, if their argument lists are different. JavaScript does not support Method Overloading.

1. **What distinguishes *object*, *class* and *interface*?**

In JavaScript every class is an object. Object is a bigger definition than class. Object can be a variable and a class can be an object to. Interface is generally a point where two systems, subjects, organizations, etc. meet and interact. A graphical user interface is a device or program enabling a user to communicate with a computer.

1. **What does *encapsulation* mean?**

Encapsulation: the action of enclosing something in or as if in a capsule.

Encapsulation in JavaScript: Like private properties, private methods may be used only by code internal to the object and are inaccessible elsewhere, whereas public methods are for outside use, and form part of the interface of the object with other code. Private properties may be accessed by means of public methods. In this sense, the private object properties are encapsulated data, because the object’s interface controls access to them.

1. **What is *polymorphism*?**

Polymorphism is one of the tenets of Object Oriented Programming (OOP). It is the practice of designing objects to share behaviors and to be able to override shared behaviors with specific ones. Polymorphism takes advantage of inheritance in order to make this happen.

1. **What does *inheritance* mean?**

In "classic OO" languages, you tend to define class objects of some kind, and you can then simply define which classes inherit from which other classes (see C++ inheritance for some simple examples). JavaScript uses a different system — "inheriting" objects do not have functionality copied over to them, instead the functionality they inherit is linked to via the prototype chain (often referred to as prototypal inheritance).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Part II: Data Structure**

# Describe the differences, if any, between *stack* and *queue*.

The stack is a collection that is based on LIFO (Last in First out), while the queue works based on FIFO (First in First out).

# In which situation should be used a *List* over an *Array* structure?

Linked lists are preferable over arrays when:

a) you need constant-time insertions/deletions from the list (such as in real-time computing where time predictability is absolutely critical)

b) you don't know how many items will be in the list. With arrays, you may need to re-declare and copy memory if the array grows too big

c) you don't need random access to any elements

d) you want to be able to insert items in the middle of the list (such as a priority queue)

Arrays are preferable when:

a) you need indexed/random access to elements

b) you know the number of elements in the array ahead of time so that you can allocate the correct amount of memory for the array

c) you need speed when iterating through all the elements in sequence. You can use pointer math on the array to access each element, whereas you need to lookup the node based on the pointer for each element in linked list, which may result in page faults which may result in performance hits.

d) memory is a concern. Filled arrays take up less memory than linked lists. Each element in the array is just the data. Each linked list node requires the data as well as one (or more) pointers to the other elements in the linked list.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# Part III: C# skills tests

**If you have no experience in C#, please move forward to the next part of the text.**

1. **What are the differences (if any) between the** System.String **and** System.Text.StringBuilder **types?**

String: String is immutable, Immutable means if you create string object then you cannot modify it and It always create new object of string type in memory.

Stringbuilder: StringBuilder is mutable, means if create string builder object then you can perform any operation like insert, replace or append without creating new instance for every time.it will update string at one place in memory doesnt create new space in memory.

1. **Please describe the three main blocks of exception handling and their usage in C# (try, catch, finalize)**
2. **Explain what the** lock **keyword is used for. i.e.:** lock (this) { ... }

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Part IV: Practice**

**At this part of the test we ask you to use any kind of programming language with which you are comfortable (C#, Java, C, C++, any pseudo language).**

I use JavaScript for the next exercises because this is what I’ve learnt. I hope you will allow it even if it is not in your list (C#, Java, C, C++, any pseudo language).

1. **Write a method that displays the first 100 Fibonacci numbers separated with commas. Example: the first 10 elements of the Fibonacci sequence are: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34 …**

**'use strict'**

**const FibonacciCalculator = (function() {**

**let fiboRow = '';**

**const oneFibonacci = function(num) {**

**if (num == 0) {**

**return 0;**

**}**

**if (num == 1) {**

**return 1;**

**}**

**else {**

**return oneFibonacci(num - 1) + oneFibonacci(num - 2);**

**}**

**}**

**const stringifier = function(number) {**

**for( let i = 0; i < number; i += 1) {**

**fiboRow += oneFibonacci(i) + ', ';**

**}**

**fiboRow = fiboRow.slice(0, -2);**

**console.log(fiboRow);**

**}**

**return {**

**stringifier: stringifier**

**}**

**})();**

**FibonacciCalculator.stringifier(100);**

1. **Create a base class named “Shape” with a method to return the area of the shape. Create a class named “Triangle” derived from the base class “Shape”. Class “Triangle” must have two constructors. One receives the length of the (*a*) side of the triangle and the related (*ma*) height. The other receives the length of (*a*),(*b*),(*c*) sides of the triangle. In the second case, the constructor must validate the input by checking that the length of one side is smaller than the sum of the lengths of the other two sides. If the input is invalid, it should throw an exception. Implement the method (available in the base class) that calculates the area of the triangle on the basis of the available data. You can use two formulas. If the length of a side and height is given: *T=(a\*ma)/2*. If the lengths of the three sides are given, you can use the Hérón formula:  
   √(〖s\*(s-a)\*(s-b)\*(s-c)〗^ ) where s=(a+b+c)/2**
2. **Write a method that can determine if a given integer number is odd or even. It is ONLY allowed to use addition or subtraction operations (do NOT use modulus ‘%’, shift ‘<<’, multiply, divide, etc.). Return true if even.**

**bool IsEven(int number)**

**'use strict'**

**const funFunFunction = function(integer) {**

**let number = integer;**

**let i = 0;**

**while (number > 0) {**

**number -= 2;**

**i += 1;**

**}**

**if (number == 0) {**

**return true;**

**}**

**if (number == -1) {**

**return false;**

**}**

**}**

1. **Write a method that can determine if a matrix has a saddle point. The input parameter is a matrix constructed of integers. The method must return a boolean value that is true if the matrix has a saddle point (an element that is the smallest in its row and the largest in its column), and false if the matrix has NO saddle point.**
2. **Write a method with an integer input parameter that must return the NEXT highest number using the same digits of the input integer.**

**i.e., if the given integer is 1234, you should return 1243.int NextHighestNumberWithSameDigits(int number)**

**You reached the end of the test. Thank you for your work!**