**Skill Test 103 - Javascript - Nodejs**

|  |  |
| --- | --- |
| Name | Esther Su |
| Date | Dec 20 |

Answer any (5) or ALL of the questions below using Node.js and email the program code logic.

|  |
| --- |
| Q1: Write a function to translate a text to Pig Latin and then back. English words are translated to Pig Latin by taking the first letter of every word, moving it to the end of the word and adding ‘ay’. “The quick brown fox” becomes “Hetay uickqay rownbay oxfay”. |
| function pigLatin = ( str ) => {  var answer = str.split( “ “).map( ( element ) ) => {  var temp = element.substr( 0, 1 );  })  return “”.concat( …answer );  }  console.log(pigLatin( “The quick bro fox” ) ); |
|  |

|  |
| --- |
| Q2: Write a program that outputs all possibilities to put + or - or nothing between the numbers 1,2,…,9 (in this order) such that the result is 100. For example 1 + 2 + 3 - 4 + 5 + 6 + 78 + 9 = 100. |
|  |

|  |
| --- |
| Q3: Write a function **on\_all** that applies a function to every element of a list. Use it to print the first twenty perfect squares. The perfect squares can be found by multiplying each natural number with itself. The first few perfect squares are 1\*1= 1, 2\*2=4, 3\*3=9, 4\*4=16. Twelve for example is not a perfect square because there is no natural number m so that m\*m=12. |
| function isSquare (n){  if ( n > 0 && Math,sqrt( n ) % 1 ===0){  return n;  }else{  return false;  }  }  console.log(isSquare(12)); |

|  |
| --- |
| Q4: Write a function that computes the list of the first 100 Fibonacci numbers. The first two Fibonacci numbers are 1 and 1. The n+1-st Fibonacci number can be computed by adding the n-th and the n-1-th Fibonacci number. The first few are therefore 1, 1, 1+1=2, 1+2=3, 2+3=5, 3+5=8. |
| function fibonacci( n ) {  var a = 1, b = 0, temp;  if ( n <= 1 ) {  return n;  } else {  while ( n >1 ){  temp = a;  a = a + b;  b = temp;  n--;  }  return b;  }  }  console.log(fibonacci(5)); |
|  |

|  |
| --- |
| Q5: Write a function that takes a list of strings an prints them, one per line, in a rectangular frame. For example the list ["Hello", "World", "from", "Massively"] gets printed as:  \*\*\*\*\*\*\*\*\*\*\*\*\*  \* Hello \*  \* World \*  \* from \*  \* Massively \*  \*\*\*\*\*\*\*\*\*\*\*\*\* |
| function printFrame( arr ) {  function fill ( str, length, char ) {  return ( str.length < length ) ? fill( str + char, length, char ) : str;  }    let size = arr.map( ( str ) => {  return str.length;  })  .reduce( ( a, b ) => {  return Math.max( a, b );  });    let line = fill( '', size + 4, '\*' );    arr = arr.map( ( element ) => {  return '\* '+ fill( element, size, ' ' ) + ' \*';  })  arr.unshift( line );  arr.push( line );    return arr.join( '\n' );;  }  console.log( printFrame( [ "Hello", "World", "in", "a", "frame" ] ) ); |

|  |
| --- |
| Q6: Write a function that rotates a list by k elements. For example [1,2,3,4,5,6] rotated by two becomes [3,4,5,6,1,2]. Try solving this without creating a copy of the list. How many swap or move operations do you need? |
| function shifter( shift, arr ){  for( var i = 0; I <shift; i++){  var buff = arr.shift();  arr.push( buff );  }  return arr;  }  console.log( shifter( 2, [ 1, 2, 3, 4, 5, 6])); |

|  |
| --- |
| Q7: Write a function that takes a list of numbers, a starting base b1 and a target base b2 and interprets the list as a number in base b1 and converts it into a number in base b2 (in the form of a list-of-digits). So for example [2,1,0] in base 3 gets converted to base 10 as [2,1]. |
|  |