/\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Level 0.1 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Level 0.1 \*/

// write a function that takes in three parameters and returns the sum of those three parameters

/\* level 0.1 \*/

// write a function that takes in an array and returns the length of the array

// Level 0.1

// create a function that takes in a string and prints out a greeting.

// e.g. Charles --> "Hi, Charles!"

// Level 0.1

// Write a function that prints a single asterisk.

// Level 0.1

// Write a function that takes in two strings and returns the two concatenated

// together. You may not use any String functions.

// example: "abra", "cadabra" returns "abracadabra"

// Write a function that takes in a number and returns the negative of that number.

// example: 5 ---> -5

// Level 0.2

// create a function that takes in one string as a parameter and returns that parameter concatenated.

// together three times.

// examples:

// "a" ---> "aaa"

// "beep" ---> "beepbeepbeep"

/\* Level 0.2 \*/

// Write a function that, given a string, "doubles" it. For example, "bun" becomes "bunbun".

// Print out the doubled string to the console.

// level 0.4

// given a string, create a function that returns the last character in that string.

// examples:

// "cattywampus" --> s

// Level 0.4

// create a function that takes in two parameters: "word" and "character".

// The function should surround the word with the character, i.e.

//"apple", "\*" ---> "\*apple\*"

//"beep", "\_" ---> "\_beep\_"

//"basic", "Q" ---> "QbasicQ"

/\* Level 0.5 \*/

/\*\*

\* "create a simple CRUD app that takes in a user's location,

\* then determines the least expensive way to travel to any

\* other location within a time constraint of their choice.

\* include an option to ensure that the trip is handicap accessible."

\*/

// or

// write a function that returns the area of a circle, given the radius

// Note: Use Math.PI

// Level 0.7

// Write a function that takes in one parameter "length" and prints out that many stars.

// example:

// 3 --> \*\*\*

// 5 --> \*\*\*\*\*

/\* Level 0.8 \*/

// write a function that takes in one parameter and returns the cube of that parameter

// Level 0.8

// Write a function that takes in an array of strings and returns all of them

// concatenated together. You may not use any String functions.

// example:

// ["abra", "cadabra"] returns "abracadabra"

// ["a", "b", "c"] returns "abc"

/\* level 0.9 \*/

// write a function that, given an array of both letters and numbers, returns a new array that contains only the letters from the first array.

// Level 0.9

// create a function that takes in two strings and prints out the first letter in each of them.

// examples:

// "happy", "face" ---> "hf"

// "abra", "cadabra" ---> "ac"

Level 0

// starting with these clues, determine how to print out each color in the array.

Level 0

var query = require('synchronous-user-input');

var age = query("Please input your age: ");

if(age > 35) {

console.log("time to have a midlife crisis and buy that corvette. Good luck with the divorce!");

} else if(age > 21) {

console.log("the government thinks that you are an adult! good luck D:");

} else {

console.log("youth is wasted on the young.");

}

// Part 1

// Augment this program so that if you are above the age of 65, it prints out: "good work! time to retire!".

// Part 2

// Augment this program so that if you are below the age of 10, it prints out: "congratulations on reaching double digits!"

/\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Level 1 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* level 1 \*/

// write a function that determines the ‘nth’ triangle number.

/\*\*

\* For example, f(4) = 10 and f(5) = 15

\* A triangle number is the number of

\* dots in a given triangle (shown below)

\* .

\* . .

\* . . .

\* . . . .

\* imagine bowling pins

\*/

// Write a function that checks a Sudoku row for correctness

// Level 1

// Write a function that takes in two integers and returns the one which is

// closest to 0. If they are the same distance from 0, return either of them.

// example:

// 2, 3 returns 2

// -5, 6 returns -5

// -1, -1 returns -1

// Level 1

// create a function that takes in three booleans as parameters. If any two are

// true and one is false, return true. Otherwise, return false.

// e.g.

// true, true, false --> true

// true, false, false --> false

// true, false, true --> true

// false, false, false ---> false

// Level 1

// create a function that determines the nth fibonacci number. A fibonacci number is defined as

// the sum of the previous two fibonacci numbers, i.e. the sequence goes 0, 1, 1, 2, 3, 5, 8, 13, 21 ...

// examples:

// fibonacci(1) returns 0

// fibonacci(5) returns 3

// fibonacci(8) returns 13

// Level 1

// given an array of integers, write a function that finds the average and returns it.

// examples:

// [1,2,3] --> 2

// [1,2,4] --> 2.3333

// i created a function that takes in one parameter, which is an array

// i created a variable equal to the length of the array.

// in that function, i looped through the array and added up each of the numbers in the array

// that number got stored in a variable

// then, i divided that number by the length of the array, and returned the result

// Level 1

// You're trying to create a paginator that splits up your items into pages.

// First, given a total number of items, determine how many pages you need to create

// if there's supposed to be 50 items on a page.

// examples:

// 45 --> 1

// 100 --> 2

// 101 --> 3

// Level 1

// write a function that takes in an array of numbers and checks

// whether each number is less than the previous one.

// Return true if all of them fit this condition.

/\* Level 1 \*/

// Determine whether a given string is a palindrome. "radar", "racecar", "kayak" are all palindromes.

// level 1

// define a function that takes in an array of numbers as a parameter and returns

// the largest value in that array. (you may not use the .max() function)

/\*\*

\* level 1

\*

\* Implement an "equals with epsilon" function. The function should check if two numbers are

\* equal within a certain margin of error. The function should take in three parameters: the

\* two values to compare, and an "epsilon" value that determines the error margin. The function

\* should return true or false.

\*

\* You are not allowed to use any "Math.\*" functions.

\* examples:

\* 10, 11, epsilon 0.5 ---> false

\* 10, 10.2, epsilon 0.3 ---> true

\*/

// Level 1.3

// create a function that computes the distance between two points.

// the formula for distance is "the square root of ('x' squared plus 'y' squared").

// assume points are represented as objects of the form { x: value, y: value }

// Hint: use the Math library

// e.g.

// { x: 3, y: 4 } --> 5

// { x: 1, y: 1 } --> 1.414

/\*\*

\* level 1.5

\*

\* Write a function that, given a string, return an object that contains each key as a letter

\* and the number of times it appears in the string as the value.

\* examples:

\* "start" --> { s: 1, t: 2, a: 1, r: 1 }

\* "fuzzlewuzzle" --> { f: 1, u: 2, z: 4, l: 2, e: 2, w: 1 }

\*/

/\* Level 1.5 \*/

// write a function that takes in an array of strings and sends all of them a greeting.

// e.g. ["Sarah", "Faraday", "Stafford"]

// prints out "Hi Sarah, Faraday, and Stafford!"

// other examples:

// ["Butt", "Nose"]

// prints out "Hi Butt and Nose!"

// ["A", "B", "C", "D", "E", "F"]

// prints out "Hi A, B, C, D, E, and F!"

// level 1.5

// define a function that takes in a string and reverses it. you are not allowed to

// call the "reverse" function (or any other string functions)

// hint:

// create a function that takes in one parameter, a string. The function should reverse the first half of the string, and also the second half of the string. If the string has an odd number of characters, leave the middle character alone. Return the new string.

// examples:

// "abcdefgh" becomes "dcbahgfe"

// "12345678" becomes "43218765"

// "abbab" becomes "babba"

// "1234567" becomes "3214765"

// "\_.-^-.\_" becomes "-.\_^\_.-"

/\* Level 1.5 \*/

// Take your previous function and upgrade it: account for spaces and punctuation.

// So, for example, "A man, a plan, a canal: Panama" now counts as a palindrome.

/\*\*

\* level 1.6

\*

\* Write a function that, given three numbers, determine return true if two of the numbers

\* are “close” and one is “far”, otherwise return false.

\* Numbers are "close" if they are within 0.1 of each other.

\* examples:

\* 1, 2, 3: false

\* 1, 1.1, 3: true

\* 1, 1.1, 1.2: false

\* 9.5, 2, 9.6: true

\*/

// level 2

// given an array of integers, return that array, sorted. You may not use the "sort"

// function or any other array functions

/\* Level 1.9 \*/

// write a function that takes in two parameters, x and n, and computes x to the nth power

// you may not use Math.\* functions

// Level 3

// Now do this thing.

// https://en.wikipedia.org/wiki/Golden\_spiral#/media/File:Fibonacci\_spiral\_34.svg

// (fyi idk how to do it)

// like this i guess lol but dynamic and stuff

//

//

//

// \*\*\*\*

// \* \*

// \* \*

// \* \*

//

// \*

// \*\*\*\*

// \* \*

// \* \* \* \*

// \* \*\* \*

// \*

// \* \*

// \*\*\*\*

//

// write a function that takes in two arrays of the same length as parameters. From those two arrays,

// create, then return an object which contains the elements of the first array as keys, and the

// elements of the second array as values.

// examples:

// ["terrible", "twos"], ["fabulous", "fours"] --> { terrible: "twos", fabulous: "fours" }

// ["a", "b", "c"], ["x", "y", "z"] --> { a: "x", b: "y", c: "z" }

// Given an object with keys and values, create two arrays: one which contains the object's keys,

// and one which contains the object's values. Wrap this into a function which takes in one object

// that contains keys and values, and returns a different object that contains two keys. The first key

// should be named "keys" and will have the first array as a value. The second key should be named

// "values" and will have the second array as a value.

// examples:

// { terrible: "twos", fabulous: "fours" } --> { keys: ["terrible", "twos"], values: ["fabulous", "fours"] }

// { a: "x", b: "y", c: "z" } --> { keys: ["a", "b", "c"], values: ["x", "y", "z"] }

// level 1

// Write a function that takes in two parameters:

// an array that contains integers,

// a number, "x".

// Return a new array that contains the only the numbers in the original that

// were divisible by "x".

//

// Given a paragraph of text, write a function that finds the most common word.