//Task 1: Applying filter, map and reduce function on arrays of numbers  
let limitFunction = function(num)  
{  
 return ((num >= 5) && (num <= 20));  
};  
  
let ***X*** = [134, 6, 7, 83, 9, 1, 0, 9, 6, 17, 54, 16];  
  
// We want to filter the elements for the array that are between 5 and 20  
// We will do that using the .filter() method  
  
let ***limitValues*** = ***X***.filter(limitFunction);  
***console***.log("the elements that are in limitValues");  
***console***.log(***limitValues***);  
  
//using the arrow notation  
//filtered values will be an array that carries all the numbers that are within the range  
// '=>' is implies/lambda expression  
  
let ***filteredValues*** = ***X***.filter(num => ((num >= 5) && (num <= 20)));  
***console***.log("the elements that are in filtered values");  
***console***.log(***filteredValues***);  
  
//////////////////////////////////////////////////////////////////////////////  
let transformToInches = function(num)  
{  
 return num\*39.37;  
};  
  
// .map() will run a callback on each element of the array  
// callback is when passing a function as an argument to another function  
  
// we wanna store the transformed numbers in another array  
// the transformToInches will be applied to the array that has the filtered values  
let ***transformedToInches*** = ***limitValues***.map(transformToInches);  
***console***.log(***transformedToInches***);  
***console***.log('\n');  
  
let ***transformedtoInches2*** = ***limitValues***.map(num => (num \* 39.37));  
***console***.log("using the arrow notation");  
***console***.log(***transformedtoInches2***);  
  
/////////////////////////////////////////////////////  
  
let findMinValue = function(a, b)  
{  
 return (a<b)?a:b; ///// if a is less than b return a, otherwise return b  
};  
  
// the findValue function will start by holding a as current value  
// it will compare it to the next value in the array  
// if b is less than a, the new current will be b, and it will compare it  
// to the next element in the array  
  
// we will use the .reduce() to find the min amongst the numbers  
  
***console***.log('\n');  
  
let ***min*** = ***transformedToInches***.reduce(findMinValue);  
***console***.log(***min***);  
  
  
let ***min2*** = ***X***.filter( num => ((num >= 5) && (num <= 20)))  
 .map(num => (num \* 39.37))  
 .reduce((a,b) => (a<b)?a:b);  
  
***console***.log(***min2***);  
  
//Task 2: Applying filter, map and reduce function on arrays of objects  
  
// we have an array of multiple objects  
// each object represents a point in the plain  
//we need to filter/exclude the points that has the x value or the y value that are equal to zero  
  
let ***points*** = [{x:5, y:6}, {x:3, y:7}, {x:8, y:0}, {x:9, y:10}, {x:15, y:4}, {x:0, y:15}];  
  
let findPointoffAxes = function(point)  
{  
 return ((point.x != 0) && (point.y != 0));  
  
};  
  
// we want to score the points that are not on any of the axes in an array  
  
let ***pointsOffAxes*** = ***points***.filter(findPointoffAxes);  
***console***.log("filtered points");  
***console***.log(***pointsOffAxes***);  
  
/////////////////////////////////////////////////////////  
  
// we want to calculate the differences of the filtered points from the origin  
// we will use the euclidean distance  
// formula is square root ( (x1-x0)^2 + (y1-y0)^2 )  
  
let findDistances = function(point)  
{  
 return(***Math***.sqrt(***Math***.pow(point.x,2) + ***Math***.pow(point.y, 2)));  
};  
  
let ***distances*** = ***pointsOffAxes***.map(findDistances);  
***console***.log('\n');  
***console***.log("Distances from the origin");  
***console***.log(***distances***);  
  
///////////////////////////////////////////////  
///////////////////////////////////////////////  
  
let findMaxDistances = function(distance1, distance2)  
{  
 return(distance1>distance2)?distance1:distance2;  
};  
  
let ***MaxDistance*** = ***distances***.reduce(findMaxDistances);  
***console***.log("\n");  
***console***.log("The maximum distances returned from using the function is:" + ***MaxDistance***);  
  
let ***maxDistance*** = ***points***.filter(point => (point.x != 0) && (point.y != 0))  
 .map( point => (***Math***.sqrt(***Math***.pow(point.x,2) + ***Math***.pow(point.y, 2))))  
 .reduce((distance1, distance2) => (distance1 > distance2)?distance1:distance2);  
  
***console***.log("the maximum distance returned from arrow notation is: "+ ***maxDistance***);