This assignment is graded based on correctness and will require you to use higher-order functions to sort automobiles. The description is below and can also be found here (Links to an external site.)Links to an external site.on jsFiddle. You should submit a single .js file called automobile.js which when run with node.js using the command "node automobile.js" produces the described results. You must make use of higher-order functions to sort the cars. You should not, for example, create entirely separate functions each with dedicated loops to sort the cars. You will need a loop (or potentially more than one loop depending on your sorting algorithm of choice) in the sortArr function but that is pretty much it. Use prototype whenever needed.

```
function Automobile( year, make, model, type ){

    this.year = year; //integer (ex. 2001, 1995)

    this.make = make; //string (ex. Honda, Ford)

    this.model = model; //string (ex. Accord, Focus)

    this.type = type; //string (ex. Pickup, SUV)

}


var automobiles = [

    new Automobile(1995, "Honda", "Accord", "Sedan"),

    new Automobile(1990, "Ford", "F-150", "Pickup"),

    new Automobile(2000, "GMC", "Tahoe", "SUV"),

    new Automobile(2010, "Toyota", "Tacoma", "Pickup"),

    new Automobile(2005, "Lotus", "Elise", "Roadster"),

    new Automobile(2008, "Subaru", "Outback", "Wagon")

    ];


/*This function sorts arrays using an arbitrary comparator. You pass it a comparator
and an array of objects appropriate for that comparator and it will return a new
array which is sorted with the largest object in index 0 and the smallest in the last
index*/

function sortArr( comparator, array ){

    /*your code here*/

}
```

```
/*A comparator takes two arguments and uses some algorithm to compare them. If the
first argument is larger or greater than the 2nd it returns true, otherwise it
returns false. Here is an example that works on integers*/

function exComparator( int1, int2){

    if (int1 > int2){

        return true;

    } else {

        return false;

    }

}



/*For all comparators if cars are 'tied' according to the comparison rules then the
order of those 'tied' cars is not specified and either can come first*/



/*This compares two automobiles based on their year. Newer cars are "greater" than
older cars.*/

function yearComparator( auto1, auto2){

    /* your code here*/

}



/*This compares two automobiles based on their make. It should be case insensitive
and makes which are alphabetically earlier in the alphabet are "greater" than ones
that come later.*/

function makeComparator( auto1, auto2){

    /* your code here*/

}



/*This compares two automobiles based on their type. The ordering from "greatest" to
"least" is as follows: roadster, pickup, suv, wagon, (types not otherwise listed). It
should be case insensitive. If two cars are of equal type then the newest one by
model year should be considered "greater".*/

function typeComparator( auto1, auto2){

    /* your code here*/

}
```

```
/*Your program should output the following to the console.log, including the opening
and closing 5 stars. All values in parenthesis should be replaced with appropriate
values. Each line is a seperate call to console.log.


Each line representing a car should be produced via a logMe function. This function
should be added to the Automobile class and accept a single boolean argument. If the
argument is 'true' then it prints "year make model type" with the year, make, model
and type being the values appropriate for the automobile. If the argument is 'false'
then the type is ommited and just the "year make model" is logged.


*****

The cars sorted by year are:

(year make model of the 'greatest' car)

(...)

(year make model of the 'least' car)


The cars sorted by make are:

(year make model of the 'greatest' car)

(...)

(year make model of the 'least' car)


The cars sorted by type are:

(year make model type of the 'greatest' car)

(...)

(year make model type of the 'least' car)

*****


As an example of the content in the parenthesis:

1990 Ford F-150 */
```