(2 points) Implement a function called compareCarsByMakeThenModel that can be passed as an argument to the compare parameter of the qksort function from the book. compareCarsByMakeThenModel should return a value that will cause qksort to sort an array of cars in ascending order (from smallest to largest) by make and, when two cars have the same make, in ascending order by model.

```
int compareCarsByMakeThenModel(const void *car1, const void *car2) {
    Car *c1 = (Car *)car1;
    Car *c2 = (Car *)car2;

    int cmp = strcmp(c1->make, c2->make);
    if (cmp == 0) {  // If makes are the same
        return strcmp(c1->model, c2->model);
    }
    return cmp;
}
```

(2 points) Implement a function called compareCarsByDescendingMPG that can be passed as an argument to the compare parameter of the qksort function from the book. compareCarsByDescendingMPG should return a value that will cause qksort to sort an array of cars in descending order (from largest to smallest) by mpg.

```
int compareCarsByDescendingMPG(const void *car1, const void *car2) {
    Car *c1 = (Car *)car1;
    Car *c2 = (Car *)car2;

    return c2->mpg - c1->mpg;  // Descending order
}
```

(2 points) Implement a function called compareCarsByMakeThenDescendingMPG that can be passed as an argument to the compare parameter of the qksort function from the book. compareCarsByMakeThenDescendingMPG should return a value that will cause qksort to sort an array of cars in ascending order by make and, when two cars have the same make, in descending order by mpg.

```
int compareCarsByMakeThenDescendingMPG(const void *car1, const void *car2) {
    Car *c1 = (Car *)car1;
    Car *c2 = (Car *)car2;

    int cmp = strcmp(c1->make, c2->make);
    if (cmp == 0) {  // If makes are the same
        return c2->mpg - c1->mpg;  // Descending order
    }
    return cmp;
}
```

(3 points) Write a program that tests your functions from parts a-c with the following array of cars:

```
int main() {
    Car cars[] = {
        {"Toyota", "Camry", 33},
        {"Ford", "Focus", 40},
        {"Honda", "Accord", 34},
        {"Ford", "Mustang", 31},
        {"Honda", "Civic", 39},
        {"Toyota", "Prius", 48},
        {"Honda", "Fit", 35},
        {"Toyota", "Corolla", 35},
        {"Ford", "Taurus", 28}
    };
    int size = sizeof(cars) / sizeof(cars[0]);

    printf("Original order:\n");
    printCars(cars, size);
    printf("\n");

    qksort(cars, size, sizeof(Car), 0, size-1, compareCarsByMakeThenModel);
    printf("Sorted by make then model:\n");
    printCars(cars, size);
    printf("\n");

    qksort(cars, size, sizeof(Car), 0, size-1, compareCarsByDescendingMPG);
    printf("Sorted by descending MPG:\n");
    printCars(cars, size);
    printf("\n");

    qksort(cars, size, sizeof(Car), 0, size-1,
compareCarsByMakeThenDescendingMPG);
    printf("Sorted by make then descending MPG:\n");
    printCars(cars, size);
    printf("\n");

    return 0;
}
```

Original unsorted cars:

Toyota Camry 33

Ford Focus 40

Honda Accord 34

Ford Mustang 31

Honda Civic 39

Toyota Prius 48

Honda Fit 35

Toyota Corolla 35

Ford Taurus 28

Cars sorted by make then model:

Ford Focus 40

Ford Mustang 31

Ford Taurus 28

Honda Accord 34

Honda Civic 39

Honda Fit 35

Toyota Camry 33

Toyota Corolla 35

Toyota Prius 48

Cars sorted by descending MPG:

Toyota Prius 48

Ford Focus 40

Honda Civic 39

Toyota Corolla 35

Honda Fit 35

Honda Accord 34

Toyota Camry 33

Ford Mustang 31

Ford Taurus 28

Cars sorted by make then descending MPG:

Ford Focus 40

Ford Mustang 31

Ford Taurus 28

Honda Civic 39

Honda Fit 35

Honda Accord 34

Toyota Prius 48

Toyota Corolla 35

Toyota Camry 33