Warning: These examples do NOT give the correct answers!

The best way to proceed is to first define two functions that implement the two equations that you will need. These two equations can then be combined in a third function that will give you the answers you seek.

**Equation 1: Braking Distance**  the first function, which I’ve called braking.distance.ft, takes a mile per hour value and computes the braking distance using an equation that I derived from the Colorado Driving Manual. A third function is defined that uses the first two to compute total stopping distance.

```r
# Equation 1
braking.distance.ft <- function(mph){0.062673 * (mph^2.5)}
```

**Equation 2: Perception-Reaction Time Distance**  The second function computes the distance in feet traveled during the given perception reaction time in seconds and the speed of the car in miles per hour. The mph is first converted to feet per second and then multiplied by the perception reaction time. In the function definition the prt is given a default value of 2 seconds.

```r
# Equation 2:
prt.distance.ft <- function(mph, prt = 2) {60 + prt * mph * 5280.0 / 3600.0}
```

Test these functions. For example, the braking distance at 60 mph should be 1747.7 feet. And the distance traveled during a 2 second perception-reaction time at 40 mph should be 177.3 feet. Remember the warning above: these equations are NOT correct.

**Equation 3: Total Stopping Distance**  The total stopping distance is the sum of the distance traveled during the perception-reaction time plus the distance traveled while breaking. Function 3 combines Function 1 and Function 2 to compute the total stopping distance given a specific speed of travel and a specific perception-reaction time. I’ve also included an additional argument, called ‘feet’ that specifies the distance of a hazard on the highway. This term is needed because in solving an equation for zero using R’s uniroot() function (see below), we need to take the hazard distance into account.

```r
# Equation 3:
total.stopping.distance.ft <- function(mph, prt = 2, feet = 0) {
  braking.distance.ft(mph) + prt.distance.ft(mph, prt) - feet
}
```

Test out equation 3: at a speed of 45 mph with a PRT of 2 seconds and no target, the total stopping distance should be 1043.4.

**Part 1 Solution**  The answer to Part 1 is given in the code below for a speed of 60 mph, with a 2 second prt.:

The answer to Part 1 is that 1983.7 feet will be required to come to a stop from 60 mph with a perception-reaction time of 2 sec.

**Part 2 Solution** There are two ways to find the answer to part 2:

1. Trial and error: try out different speeds in Function 3 to find the value that gives 210 feet as the answer;

2. Use the univariate root finder (uniroot()) to find the speed (mph) that will bring the returned value of Function 3 to zero with an input value of the “feet” argument given a value of 120 feet (low-beam condition) and another solution for 250 feet (high beam condition). Check out the help for uniroot() to learn more about this method.

```r
cmaxStoppingDist.low <- 120
c.low <- uniroot(total.stopping.distance.ft, c(1, 100),prt = 2, feet = maxStoppingDist.low)

Low Beam Solution

cmaxStoppingDist.high <- 250
c.high <- uniroot(total.stopping.distance.ft, c(1, 100),prt = 2, feet = maxStoppingDist.high)

High Beam Solution

The solutions for the low- and high-beam condition are shown below:

```r
print(paste("root low beam = ", round(c.low$root, 1)))
## [1] "root low beam = 11.3"

print(paste("root high beam = ", round(c.high$root, 1)))
## [1] "root high beam = 21.1"

**Plotting the graph** Plot the stopping distance for a range of values. Note that I have used the answers for the low- and high-beam speeds to draw the horizontal and vertical lines. Of course you need to discover what that value is by using uniroot and a distance value of either 120 feet (low beam) or 250 feet (high beam). I have made this graph rather fancy, using the text command to insert various text and numeric values in the graph. Such annotation can greatly improve most graphs, as long as you don’t make ti too cluttered.
# first plot the stopping distance for different speeds
x <- seq(1, 60, 0.1)  # generate a range of speeds from 1 to 60 mph
y <- total.stopping.distance.ft(x, 2)  # compute the stopping distance for each x

plot(y ~ x, type="l", lwd=2,
     xlab = "Car Speed in mph",
     ylab = "Total Stopping Distance in Feet",
     main = "Maximum Safe Speed for Low- and High-Beams")

# add the lines for low beams
low.spd <- z.low$root  # max speed for low beams
low.dst <- maxStoppingDist.low  # illumination distance with low beams
lines(c(-10, low.spd, low.spd), c(low.dst, low.dst, 0),
     lty = "dashed", col = "red")
text(low.spd, -4,
     paste(round(low.spd, 1), "mph"), cex=0.7)
text(10, low.dst,
     paste(round(low.dst, 1), "feet visibility with low beams"),
     pos = 3, cex=0.7)

# add the lines for high beams
high.spd <- z.high$root
high.dst <- maxStoppingDist.high
lines(c(-10, high.spd, high.spd), c(high.dst, high.dst, 0),
     lty="dashed", col="red")
text(high.spd, -4,
     paste(round(high.spd, 1), "mph"), cex=0.7)
text(10, high.dst,
     paste(round(high.dst, 1), "feet visibility with high beams"),
     pos = 3, cex=0.7)
**Maximum Safe Speed for Low- and High-Beams**

- **11.3 mph**: 120 feet visibility with low beams
- **21.1 mph**: 250 feet visibility with high beams