Using ggplot: Answers

1. **Load the ggplot2 library (install it if you have to) and the diamonds dataset using data()**

```
# if needed install.packages("ggplot2")
library(ggplot2)
data("diamonds")
```

2. **Explore the dataset using dim(), str() and help(), which variables are continuous, which variables are discrete? Is this dataset ready for plotting with ggplot?**

```
dim(diamonds)
str(diamonds)
help(diamonds)
```

Cut, color, and clarity are factors, and therefore discrete. The others are numeric continuous variables.

3. **Use ggplot to plot a scatterplot of the relationship between the diamonds' carat and their price**

```
ggplot() + geom_point(data=diamonds, aes(x=carat, y=price))

#or

ggplot(diamonds) + geom_point(aes(x=carat, y=price))

#or

ggplot(diamonds, aes(x=carat, y=price)) + geom_point()
```

4. **Make all dots darkblue and set the alpha value to 0.1**

```
ggplot(diamonds, aes(x=carat, y=price)) + geom_point(color="darkblue", alpha=0.1)
```

5. **Visualize the influence of the color of a diamond on its price by mapping the diamond color to the color aesthetic**

```
#The color of the dots will be overwritten if we specify it statically
#in the geom_point function itself
ggplot(diamonds, aes(x=carat, y=price, color=color)) + geom_point(alpha=0.1)
```

6. **Use a ggplot barplot to visualize diamond clarity depending on color, map diamond color to x and diamond clarity to fill**

```
ggplot(diamonds, aes(x=color, fill=clarity)) + geom_bar()
```

7. **Create a boxplot of the carat of a diamond based on its clarity and add whiskers using stat_boxplot**

```
ggplot(diamonds, aes(x=clarity, y=carat)) +
    stat_boxplot(geom="errorbar", width=0.5) +
    geom_boxplot()
```

8. **Add a geom_point layer to the previous plot mapping the diamonds price to the color**

```
ggplot(diamonds, aes(x=clarity, y=carat)) +
    stat_boxplot(geom="errorbar", width=0.5) +
    geom_boxplot() +
    geom_point(aes(color=price))
```

9. **Create a histogram of the price of the diamonds and separate the histograms into facets using diamond color, choose a good binwith or number of bins**

```
ggplot(diamonds, aes(x=price)) +
    geom_histogram(binwidth = 100) +
    facet_grid(color ~ .)
```

10. **Create a grid of facets of the same histogram by comparing both color and cut**

```
ggplot(diamonds, aes(x=price)) +
    geom_histogram(binwidth = 100) +
    facet_grid(color ~ cut)
```

11. **Use 'aggregate(diamonds, by = list(cut = diamonds$cut, color = diamonds$color), mean)' to calculate the mean of all variables by cut and color. Create a heatmap of the mean prices by cut and color using geom_tile**

```
#Aggregate uses a function (in this case mean) to aggregate all variables
#in a given data.frame by a list of variables given in "by".
#Note: excluding columns 2,3,4 since they are factors
#and therefore don't have means
mean.price <- aggregate(diamonds[, -c(2,3,4)],
    by = list(cutAgg = diamonds$cut, colorAgg = diamonds$color),
    mean)
ggplot(mean.price, aes(x=cutAgg, y=colorAgg, fill=price)) +
    geom_tile()
```

12. **Change the title of the heatmap to "Average prices"**

```
ggplot(mean.price, aes(x=cut, y=color, fill=price)) +
    geom_tile() +
    labs(title="Average prices")
```

13. **Change the gradient of the fill scale using 'scale_fill_gradient2'. Have it go from darkblue to white to darkred, set the midpoint to 4500**

```
ggplot(mean.price, aes(x=cut, y=color, fill=price)) +
    geom_tile() +
    labs(title="Average prices") +
    scale_fill_gradient2(low="darkblue", mid="white", high="darkred", midpoint = 4500)
```

14. **Choose and add a theme to the heatmap, or create a theme yourself using the options listed at http://ggplot2.tidyverse.org/reference/theme.html**

```
ggplot(mean.price, aes(x=cut, y=color, fill=price)) +
    geom_tile() +
    labs(title="Average prices") +
    scale_fill_gradient2(low="darkblue", mid="white", high="darkred", midpoint = 4500) +
    theme_minimal()
```