```
#
# DS5 Exercise to show how to use gather, separate, and unite to tidy
# data when data is spread across multiple columns
#
library(tidyverse)

filename <- "leaftc.csv"
d <- read_csv(filename)

select(d, 1:5)

# When we look at this table, we can see that it is in y format.
# Also note that there are values for two variables with the column
# names encoding which column represents which variable.

# A tibble: 2 x 5
# country `1960_fertility` `1960_life_expe~ `1961_fertility` `1961_life_exp~
# <chr>              <dbl>            <dbl>            <dbl>           <dbl>
# Germany             2.41             69.3             2.44            69.8
# South ~             6.16             53               5.99            53.8

# We can start the data wrangling with the gather function,
#but we should no longer use the column name Year for the new columns,
# since it also contains the variable type. We will call it key.
#That's the default of this function. So we write this piece of code
#to gather the data.

dat <- d %>% gather(key, value, -country)
head(dat)

# A tibble: 6 x 3
# country     key                  value
# <chr>       <chr>                <dbl>
# Germany     1960_fertility        2.41
# South Korea 1960_fertility        6.16
# Germany     1960_life_expectancy 69.3
# South Korea 1960_life_expectancy 53
# Germany     1961_fertility        2.44
# South Korea 1961_fertility        5.99

# The first challenge to achieve this is to separate the key column
# into the year and the variable type. We can add a third column to catch this and let
# the separate function know which column to fill in with missing values--
# NAs, in this case--  when there is no third value.

dat %>% separate(key, c("year", "first_variable_name", "second_variable_name"), "_", fill = "right")

# However, if we read the separate file, we find that a better approach
# is to merge the last two variables when there's an extra separation
# using the argument extra, like this.

dat %>% separate(key, c("year", "variable_name"), sep = "_", extra = "merge")

# However, we're not done yet. We need to create a column for each
# variable. As we've learned, the spread function can do this. So now,
# to create tidy data, we're actually using the spread function.
# So we write this piece of code, and when we run it,

dat %>% separate(key, c("year", "variable_name"), sep = "_", extra = "merge") %>% spread(variable_name, value)

# we now get a fertility and a life expectancy column. Alternatively, we
# could use separate and unite like this:

dat %>% separate(key, c("year", "first_variable_name", "second_variable_name"), "_", fill = "right") %>%
unite(variable_name, first_variable_name, second_variable_name, sep = "_")

# A tibble: 10 x 4
# country     year  variable_name   value
# <chr>       <chr> <chr>           <dbl>
# Germany     1960  fertility_NA     2.41
# South Korea 1960  fertility_NA     6.16
# Germany     1960  life_expectancy 69.3
# South Korea 1960  life_expectancy 53
# Germany     1961  fertility_NA     2.44
# South Korea 1961  fertility_NA     5.99
# Germany     1961  life_expectancy 69.8
# South Korea 1961  life_expectancy 53.8
# Germany     1962  fertility_NA     2.47
# South Korea 1962  fertility_NA     5.79

# ...and then spread the columns with this code:
dat %>%
  separate(key, c("year", "first_variable_name", "second_variable_name"),"_", fill = "right") %>%
  unite(variable_name, first_variable_name, second_variable_name, sep = "_") %>%
  spread(variable_name, value)
```

```
# A tibble: 6 x 4
# country     year  fertility_NA life_expectancy
# <chr>       <chr>        <dbl>           <dbl>
# Germany     1960          2.41            69.3
# Germany     1961          2.44            69.8
# Germany     1962          2.47            NA
# South Korea 1960          6.16            53
# South Korea 1961          5.99            53.8
# South Korea 1962          5.79            NA
```