In [1]:

```
library(tidyverse)
library(nycflights13)
```

```
── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──
✔ ggplot2 2.2.1     ✔ purrr   0.2.4
✔ tibble  1.4.1     ✔ dplyr   0.7.4
✔ tidyr   0.7.2     ✔ stringr 1.2.0
✔ readr   1.1.1     ✔ forcats 0.2.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

STATS 306

Problem set 2: data manipulation

Each question is worth two points, for a total of 20. For each of these questions, start with the `flights` table in the `nycflights13` data set.

Problem 1

Use the `%in%` operator to extract all the flights that took place in months that begin with the letter "J". Store the resulting table in a variable called `table1`.

In [2]:

```
table1 = NA
### BEGIN SOLUTION
table1 = filter(flights, month %in% which(substr(month.name,0,1) == "J"))
### END SOLUTION
```

In [3]:

```
stopifnot(exists("table1"))
### BEGIN HIDDEN TESTS
stopifnot(identical(table1, filter(flights, month %in% which(substr(month.name,0,1) == "J"))))
### END HIDDEN TESTS
```

Problem 2

Find the number of United Airlines flights that departed exactly on time and arrived exactly on time. Store this number in a variable called n2.

In [4]:

```
n2 = NA
### BEGIN SOLUTION
n2 = nrow(filter(flights, carrier=="UA", dep_time==sched_dep_time, arr_time==sched_arr_time))
### END SOLUTION
```

In [5]:

```
stopifnot(exists("n2"))
### BEGIN HIDDEN TESTS
n2_ans = nrow(filter(flights, carrier=="UA", dep_time==sched_dep_time, arr_time==sched_arr_time))
stopifnot(n2 == n2_ans)
### END HIDDEN TESTS
```

Problem 3

Find all flights destined for LAX or SF0 which departed more than an hour late, but did not arrive late. Store this table in a variable called `table3`.

In [6]:

```
table3 = NA
### BEGIN SOLUTION
table3 = filter(flights, dest %in% c("LAX","SFO"),
                dep_delay>60, arr_delay<=0)
### END SOLUTION
```

In [7]:

```
stopifnot(exists("table3"))
### BEGIN HIDDEN TESTS
table3_ans = filter(flights, dest %in% c("LAX","SFO"), dep_delay>60, arr_delay<=0)
stopifnot(identical(table3, table3_ans))
### END HIDDEN TESTS
```

Problem 4

Find the number of flights in May that are missing their departure time or their arrival time, but not both. Store this number in a variable called n4.

In [8]:

```
n4 = NA
### BEGIN SOLUTION
n4 = nrow(filter(flights, month==5, xor(is.na(dep_time), is.na(arr_time))))
### END SOLUTION
```

In [9]:

```
stopifnot(exists("n4"))
### BEGIN HIDDEN TESTS
n4_ans = nrow(filter(flights, month==5, xor(is.na(dep_time), is.na(arr_time))))
stopifnot(n4 == n4_ans)
### END HIDDEN TESTS
```

Problem 5

There are four flights which have a tail number that does not begin with the letter "N". Find those rows and store them in a variable called `table5`. (*Hint*: see the help for the `str_sub()` command.)

In [10]:

```
table5 = NA
### BEGIN SOLUTION
table5 = filter(flights, str_sub(tailnum,0,1) != "N")
### END SOLUTION
```

In [11]:

```
stopifnot(exists("table5"))
### BEGIN HIDDEN TESTS
table5_ans = filter(flights, str_sub(tailnum,0,1) != "N")
stopifnot(identical(table5, table5_ans))
### END HIDDEN TESTS
```

Problem 6

Find the tail number of the flights that were in the air for the shortest and longest amounts of time. Store them in variables `shortest_tail_num` and `longest_tail_num`, respectively.

In [12]:

```
shortest_tail_num = NA
longest_tail_num = NA
### BEGIN SOLUTION
shortest_tail_num = arrange(flights, air_time)$tailnum[c(1,2)]
longest_tail_num = arrange(flights, desc(air_time))$tailnum[1]
### END SOLUTION
```

In [13]:

```
stopifnot(exists("shortest_tail_num"))
stopifnot(exists("longest_tail_num"))
### BEGIN HIDDEN TESTS
stopifnot(shortest_tail_num %in% filter(flights, air_time == min(air_time, na.rm=T))$tailnum)
stopifnot(longest_tail_num %in% filter(flights, air_time == max(air_time, na.rm=T))$tailnum)
### END HIDDEN TESTS
```

Problem 7

Sort the rows of `flights` such that the months are arranged in the following order: spring, summer, fall, winter. (Here we define winter to be January-March, spring is April-June, etc.) Within each season, the months should be sorted in ascending order. After sorting, drop all columns except for month, day and tail number. Store the sorted and subsetted table in a variable called `table7`.

In [14]:

```
table7 = NA
### BEGIN SOLUTION
flights$shifted_month = (flights$month - 4) %% 12
table7 = arrange(flights, shifted_month, month) %>% select(month, day, tailnum)
### END SOLUTION
```

In [15]:

```
stopifnot(exists("table7"))
### BEGIN HIDDEN TESTS
flights$shifted_month = (flights$month - 4) %% 12
table7_ans = arrange(flights, shifted_month, month) %>% select(month, day, tailnum)
stopifnot(identical(table7$month, table7_ans$month))
stopifnot(identical(colnames(table7), colnames(table7_ans)))
### END HIDDEN TESTS
```

Problem 8

Drop the even-numbered columns of `flights`. Store the resulting data table in a variable called `table8`.

In [16]:

```
table8 = NA
### BEGIN SOLUTION
table8 = select(flights, -seq(2, ncol(flights), 2))
### END SOLUTION
```

In [17]:

```
stopifnot(exists("table8"))
### BEGIN HIDDEN TESTS
table8_ans = select(flights, -seq(2, ncol(flights), 2))
stopifnot(identical(table8, table8_ans))
### END HIDDEN TESTS
```

Problem 9

Define a function pick_columns which accepts a vector v and returns a data table consisting of all of the columns in `flights` whose names are found in v. The skeleton of the function has been provided for you.

In [18]:

```
pick_columns = function(v) {
    ### BEGIN SOLUTION
    select(flights, one_of(v))
    ### END SOLUTION
}
```

In [19]:

```
stopifnot(exists("pick_columns"))
### BEGIN HIDDEN TESTS
v = sample(colnames(flights), 4)
stopifnot(identical(pick_columns(v), select(flights, one_of(v))))
# select(flights, v) also works
### END HIDDEN TESTS
```

Problem 10

Select all the columns in flights which end with "delay". In the resulting table, define a new column `max_delay` which equals the maximum of the departure and arrival delays for each flight. For example, if a flight had a departure delay of `-1` and an arrival delay of `10`, the `max_delay` would equal `10`. Store this table in a variable called `table10`.

In [20]:

```
table10 = NA
### BEGIN SOLUTION
# pmax() is the easiest solution
table10 = select(flights, ends_with("delay")) %>%
                 mutate(max_delay=ifelse(arr_delay > dep_delay,
                                         arr_delay,
                                         dep_delay))
### END SOLUTION
```

In [21]:

```
stopifnot(exists("table10"))
### BEGIN HIDDEN TESTS
table10_ans = select(flights, ends_with("delay")) %>%
              mutate(max_delay=pmax(arr_delay, dep_delay))
stopifnot(identical(table10, table10_ans))
### END HIDDEN TESTS
```