Daniel Hanna
Assignment 1

Assignment 1: tcnjParkingSpaces

VM Path: `/home/student1/tcnjParkingSpaces`

Directory Contents:

```
Assignment 1.pdf                tcnjParkingSpaces.rb
historical_parking.csv          parking_lot_contraints.csv
readme.pdf
```

Instructions for running the program:

1. If using personal machine, ensure that ruby is installed on local system
    a. This can be checked by entering `ruby -v` into the machine's terminal.
    b. If an error occurs for the above command, install ruby onto the machine.

    Note: The virtual machine that hosts these files already has ruby installed; if connected to virtual machine, skip to step 2.

2. Once ruby has installed successfully, navigate to the directory where the project folder is saved, and enter the project folder to where the `tcnjParkingSpaces.rb` file is saved.
    a. This can be done by using the `cd` command to enter directories and `ls` to list the directories contents.

    Note: The file can also be run without navigating to the specific directory (see step 3).

3. Once in the project folder, enter the following command: `ruby tcnjParkingSpaces.rb`
    a. If warnings should be displayed, enter the following command:
    `ruby -w tcnjParkingSpaces.rb`

    Note: The file can also be run with the following command, if navigation to the project's directory is not desired: `ruby <file_directory>/tcnjParkingSpaces.rb`

4. The program will run. Follow the prompts on the screen closely.

5. When asked for the CSV files:
    a. Input the CSV file names only if the files are in the same directory as the ruby file
    b. If the CSV files are in a different directory, that directory will have to be inputted according to the format specified by the prompt.

Known Bugs, Issues, Limitations

1. The program considers the CSV files by reading each file an mapping each row of the file's data to a hash table. Since this operation is done row by row, a file with a large number of rows will result in the program being delayed for longer.
    a. For example, the program's reading the lot constraints csv file is nearly instantaneous (the file is roughly 20 or so rows), whereas the historical csv file that is provided takes a few seconds to read (the file is roughly 197,000 rows).
        i. Speed will also depend on how powerful of a machine the program is being run on.

2. The program estimates the best lot for parking by taking in the user's date and comparing that to the same date 1 year ago in the historical lots file.
    a. If the user enters a date who's 1-year-back conversion does not have a match in the historical lot file, the program will enter an infinite loop of attempting to change the user's date by 1 year in order to match it to a date in the historical file (which does not exist). The program may exit the loop abnormally, or . will execute the loop without terminating.
        i. It is assumed that the user is using this program for a practical use case, i.e they are typing in a current (or future) date to see the best place to park on campus. As a result, a "previous date" check is not executed in the program.
    b. This program also assumes that a complete historical dataset is present.
        i. The program will still "work" if the historical data only has data for 2 days for example, but it will enter an infinite loop if the user does not enter one of these two days for their current year.

3. The program asks for the user to enter the day of the week, but also asks for the day of the month as well.
    a. Day of the week is used to compare user's desired parking day, time, and title in comparison to the lot constraints, which have specific permissions for specific days of the week. The day of the month is used in order to find a similar date in the historical lot data, to recommend the best place to park.

4. The program does not "restart" to allow user to test multiple permissions/times/dates.