Tallest Buildings Exercise

The data in the file `tallest_bldgs.txt` contains information on the world's 200 tallest buildings as of the year 2010. The variables in this file are: `bldg_name, city, country, year, stories, height_m`

- bldg_name: building name
- city: city in which the building is located
- country: country in which building is located
- stories: the number of stories
- year: the year in which the building was structurally completed.
- height_m: height in meters

1. From the command window, import the data in the file 'tallest_bldgs.txt' and save it to a table named buildings.

```
>> buildings = readtable('tallest_bldgs.txt');
```

2. The dot notation (tableName.VariableName) may be used to create a MATLAB column vector from a table variable. Create a numeric vector named height_feet which contains the heights of all the buildings converted into feet. (1 meter = 3.28084 feet)

```
>> height_feet = buildings.height_m * 3.28084;
```

3. Modify the existing buildings table to include an additional variable called height_m at the end containing the height data you just calculated. Notice the use of the curly brackets, {} as one way to add a variable to a table. The curly brackets are also a good way to work with a subset of a table.

```
>> buildings{:,'height_feet'} = height_feet;
```

4. The dot notation is simplest when working with all of the data from a table variable. Remove the height_m table variable.

```
>> buildings.height_m = [];
```

5. The sorting capability is a good reason for using tables to hold data. Sort the values in the buildings table in order of decreasing height.

```
>> buildings = sortrows(buildings, 'height_feet', 'descend');
```

6. Indexing a table using parenthesis can create a table from a portion of the original table. If curly brackets were used, a vector or matrix would be created from the table data. Create a table that contains the data of the five tallest buildings.

```
>> fiveTallest = buildings(1:5,:)
```

7. Write the contents of `fiveTallest` to a file named '`tallBldgs.txt`'.

```
>> writetable(fiveTallest, 'tallBldgs.txt');
```

8. Create a logical vector of the buildings over 1000 feet tall.

```
>> over1k = buildings.height_feet > 1000;
```

9. Find the number of buildings that are over 1000 feet tall. Store the result in n1k.

```
>> n1k = nnz(over1k);
```

10. Create a table of the buildings over 1000 feet tall.

```
>> tallest = buildings(over1k,:);
>> tallest(1:5,:)
>> tallest(end-4:end,:)
```

11. Sort the tallest buildings by age. The default sorting order is ascending.

```
>> oldtall = sortrows(tallest,'year');
>> oldtall(1:5,:)
```

12. Using the table dot notation, table variables may be used column vectors with results saved to a new table variable. Determine which buildings have the most and least head room on each floor (story).

```
>> buildings.feet_per_story = buildings.height_feet./buildings.stories;
>> buildings = sortrows(buildings, 'feet_per_story', 'descend');
>> buildings(1:5,:)
>> buildings(end-4:end,:)
```