```
% J Hundley
% assign05a.m
%{
   You can use trigonometry to find the height of a building.
   Suppose you measure the angle between the line of sight and the
   horizontal line connecting the measuring point and the building.
   You can calculate the height of the building with the following formulas:
        tan(theta) = height / distance
        height = distance * tan(theta)
   Assume that the distance to the building along the ground is 120 meters
   and the angle measured alone the line of sight is 30 degrees
   plus/minus 3 degrees.
   --Create a 15x1 vector of random numbers bound by the 5-45 degrees.
   --"find" the angles within the target angle range, 27-33  (p.278)
a) compute the height of the building for given the distance and each acceptable angle
   display a label and the table of angles and heights from the results of a).
b) get average angle in the vector
   if average theta not with range print message too small or too large
      else compute and display average and height
%}

clc, clear all
format bank
%*****CONSTANT*****
MAX_RAND_ANGLE   =  45;  % degrees broarder angle measured along line of sight
MIN_RAND_ANGLE   =  5;   % degrees
MAX_TARGET_ANGLE =  33;  % degrees target angle measured along line of sight
MIN_TARGET_ANGLE =  27;  % degrees
DISTANCE         = 120;  % meters  distance to the building along the ground

%*****INPUT*****
% create a 15 x 1 vector of angles within the broader given angle range
randNum = rand(15,1);
theta   = ( MAX_RAND_ANGLE - MIN_RAND_ANGLE ) * randNum + MIN_RAND_ANGLE

% find the angles within the target angle range (p.282)
element = find( theta >= MIN_TARGET_ANGLE & theta <= MAX_TARGET_ANGLE );

%*****COMPUTATION*****
% compute the height of the building for given distance and each acceptable angle
heightAll = DISTANCE .* tand( theta(element) );

% get overall average angle of ALL angles in random vector
meanTheta  = mean( theta );
% compute the height of the building using the average angle
heightMean = DISTANCE .* tand( meanTheta );

%*****OUTPUT*****
% if there are angles within target range,
%    print title, headers and table of angles within the original range and heights
if ~isempty( element )
    disp('         Angle         Height')
    disp('        Degrees        Meters')
    disp( [ theta(element),heightAll ] )
end

% if overall average angle not within target range print message too small or too large
%    else compute and display average and height
if meanTheta < MIN_TARGET_ANGLE
    disp( [ num2str(meanTheta), ' degrees, average angle too small' ] )
elseif meanTheta > MAX_TARGET_ANGLE
    disp( [ num2str(meanTheta), ' degrees, average angle too large' ] )
else
    disp( [ 'The height is ', num2str(heightMean), ' meters for the average angle ',...
        num2str(meanTheta),  ' degrees.' ] );
end
```

```
% J Hundley
% assign05b.m
%{
   You can use trigonometry to find the height of a building.
   Suppose you measure the angle between the line of sight and the
   horizontal line connecting the measuring point and the building.
   You can calculate the height of the building with the following formulas:
        tan(theta) = height / distance
        height = distance * tan(theta)
   new - wrap your assign05a.m statements in a counting "for" loop
        ask the user how many times to repeat the assign05a requirements
%}

clc, clear all
format bank
%*****CONSTANT*****
MAX_RAND_ANGLE   =  45;  % degrees broader angle measured along line of sight
MIN_RAND_ANGLE   =  5;   % degrees
MAX_TARGET_ANGLE =  33;  % degrees target angle measured along line of sight
MIN_TARGET_ANGLE =  27;  % degrees
DISTANCE         = 120;  % meters  distance to the building along the ground

%*****INPUT*****
% ask user for number of times to run the assign05a requirements
numRuns = input( 'Enter the number of times to repeat the assign05a requirements: ' );
disp( ' ' )
% for each run
for r = 1 : numRuns
    disp( [ 'For run #' num2str(r) ] )

    % create a 15 x 1 vector of angles within the broader given angle range
    randNum = rand(15,1);
    theta   = ( MAX_RAND_ANGLE - MIN_RAND_ANGLE ) * randNum + MIN_RAND_ANGLE;

    % find the angles within the target angle range (p.282)
    element = find( theta >= MIN_TARGET_ANGLE & theta <= MAX_TARGET_ANGLE );

    %*****COMPUTATION*****
    % compute the height of the building for given the distance and each acceptable angle
    heightAll = DISTANCE .* tand( theta(element) );

    % get overall average angle of all angles in random vector
    meanTheta  = mean( theta );
    % compute the height of the building using the average angle
    heightMean = DISTANCE .* tand( meanTheta );

    %*****OUTPUT*****
    % if there are angles within target range,
    %    print title, headers and table of angles within the original range and heights
    if ~isempty( element )
        disp('         Angle         Height')
        disp('        Degrees        Meters')
        disp( [ theta(element),heightAll ] )
    end

    % if overall average angle not within target range print message too small or too large
    %    else compute and display average and height
    if meanTheta < MIN_TARGET_ANGLE
        disp( [ num2str(meanTheta), ' degrees, average angle too small' ] )
    elseif meanTheta > MAX_TARGET_ANGLE
        disp( [ num2str(meanTheta), ' degrees, average angle too large' ] )
    else
        disp( [ 'The height is ', num2str(heightMean), ' meters for the average angle ',...
            num2str(meanTheta),  ' degrees.' ] );
    end

    disp( ' ' )

end % end for loop
```

Read all instructions before beginning your work.

COMP1200-MatLab - assign 05
Due 4:45 pm – Wednesday – October 9, 2019
Submit `assign05a.m` and `assign05b.m` **via Canvas**

NOTE:
Your submitted file(s) MUST be spelled and cased as instructed.
[-5 points for not doing so.]

Before you start writing your program:
Read the complete instructions.

DUE Wednesday, Oct. 9

***Problem*:**
Trigonometry can be used to find the height of a building. Suppose you measure the angle between the line of sight and the horizontal line connecting the measuring point and the building. You can calculate the height of the building with the following formulas:

`tan(theta)=h/d` `h=d*tan(theta)`

Assume that the distance to the building along the ground is 120 meters and the angle measured along the line of sight is 30 degrees plus/minus 3 degrees.

***Program*: assign05a.m**

Using the requirements below, **create an algorithm**. **Type your algorithm as comments** in your script and use them as a guide as you type statements to accomplish the requirements.
START YOUR SCRIPT WITH THE REQUIRED COMMENTS GIVEN IN ASSIGNMENTS 2-4. See below.

***** CONSTANTS *****

***** INPUT *****
Create a 15x1 vector of random degrees bound by the given limits, 5-45 degrees.
`find()` the angles in the above 15x1 vector that are within the **target angle range**.

***** COMPUTE *****
Compute the heights of the building for given the distance and each angle that is within the **target angle range**.
Use the `find()` results as the index
Determine the **overall average angle** for the **15x1 vector of angles**.
Compute the height of the building using the **overall average angle**.

***** OUTPUT *****
Did `find()` output loactions of angles within the **target angle range**, i.e. is the resulting vector empty?
If there are values within the **target angle range**, display a table of angle and heights with column headers; otherwise do not print the table.
If the **overall average angle** is not within **target angle range**, display message too small or too large; otherwise display the **overall average angle** and the height computer using the **overall average angle** (from a above).

Do not use commands and statements beyond what has been taught on class.

New commands:
`find()`
`mean()`
using a vector as an index
`tand()`
`isempty()` or `~isempty`
`if..elseif..else`

Continue to use:
`format bank, compact`
`rand()`
`disp()`
still not fprintf()

***Problem Constants*:**
```
MAX_RAND_ANGLE =  45; % degrees **
MIN_RAND_ANGLE =   5; % degrees **
MAX_ANGLE      =  33; % degrees *
MIN_ANGLE      =  27; % degrees *
DISTANCE       = 120; % meters
```
***Problem Inputs*:**
15x1 vector containing angles (theta) in degrees for given broader** range
Problem Outputs:
heights of the building for given the distance and angles within the **target angle range***
overall average angle
height of the building using the **overall average angle**
Other variables:
results from `find();` the location of the angles in the 15x1 vector that are within the **target angle range***
Equations:
See above.

Sample Output:

Run 1:

```
             Angle         Height
           Degrees         Meters
            30.85          71.68
            32.19          75.53
            31.20          72.69
            28.41          64.91

    25.1219 degrees, average angle too small
```

Run 2:

```
    23.6973 degrees, average angle too small
```

Run 3:

```
             Angle         Height
           Degrees         Meters
            31.23          72.76
            32.15          75.42
            31.22          72.73

    The building height is 67.2796 meters for the average angle 29.2778 degrees.
```

***Program*: assign05b.m**

Save your assign05a.m as assign5b.m.
Edit comments as needed.
Add a `for` loop (counting loop) that will allow you to have multiple runs without restarting your script.
Add blank lines between output information.

***Problem Inputs*:**
number of runs
control variable for loop

Do not use commands and statements beyond what has been taught on class.

New commands:
`for` loop, a counting loop

Continue to use:
all from assign05a.m
Use descriptive variables.

Sample Output:

```
Enter the number of times to repeat the assign05a requirements: 3

For run #1
21.1703 degrees, average angle too small

For run #2
         Angle         Height
       Degrees         Meters
        28.80          65.96
        29.11          66.83

21.6782 degrees, average angle too small

For run #3
         Angle         Height
       Degrees         Meters
        28.14          64.18

24.6059 degrees, average angle too small
```

Instructions for all assignment scripts:

- ☐ See Standards for Documentation of MATLAB Programs on the Canvas Resources page.
- ☐ Insert comments at the top and throughout each file.
  - o Include the follow comments at the beginning of this (and ALL) files.
    - `%` submitter's name, **GROUP # or "none"**
    - `%` other group members' names or "none"
    - `%` **`program file name`**, ex. assign02a.m
    - `%` due date of the assignment
    - `%` **statement about collaboration REQUIRED.**
    - `%` a short narrative about what the file does
  - o Use the algorithm given as comments throughout your program.
- ☐ Observe the instructor's rule for naming variables.
  - o Use ALL CAPS for constants variable names.
  - o Start other variables with lower case.
  - o Use descriptive variable names.
- ☐ Use Sample Input/Output as a guide.
- ☐ Code clarity:
  - o Indent blocks as needed. **Use Smart Indent.**
  - o Divide your solution program code into sections as noted in the algorithm. Use blank lines as needed to group statements.
  - o Use section comments as well as the algorithm step comments.
  - o Remove statements from previous assignments that do not apply to the current requirements.
- ☐ Use comments to show units.
- ☐ **Use the CONSTANT and variable names, not numbers.**
  Exceptions are incrementers (or counters) and numbers without identity.
- ☐ No extra output, i.e. use semicolons

GRADE OF ZERO for a file if submitter name not part of Canvas group.

(-3pts) No CURRENT GROUP# or "none".
(-3pts) For your own protection, type "none" for other group members if submitting alone.
(-5pts) Five point penalty for not joining your Canvas group.

(-5pts) Zero points for comments if no collaboration statement.

Submit via Canvas:

`assign05a.m`	MATLAB script file
`assign05b.m`	MATLAB script file

NOTE:
Your submitted file(s) MUST be spelled and cased as instructed.
[-5 points for not doing so.]