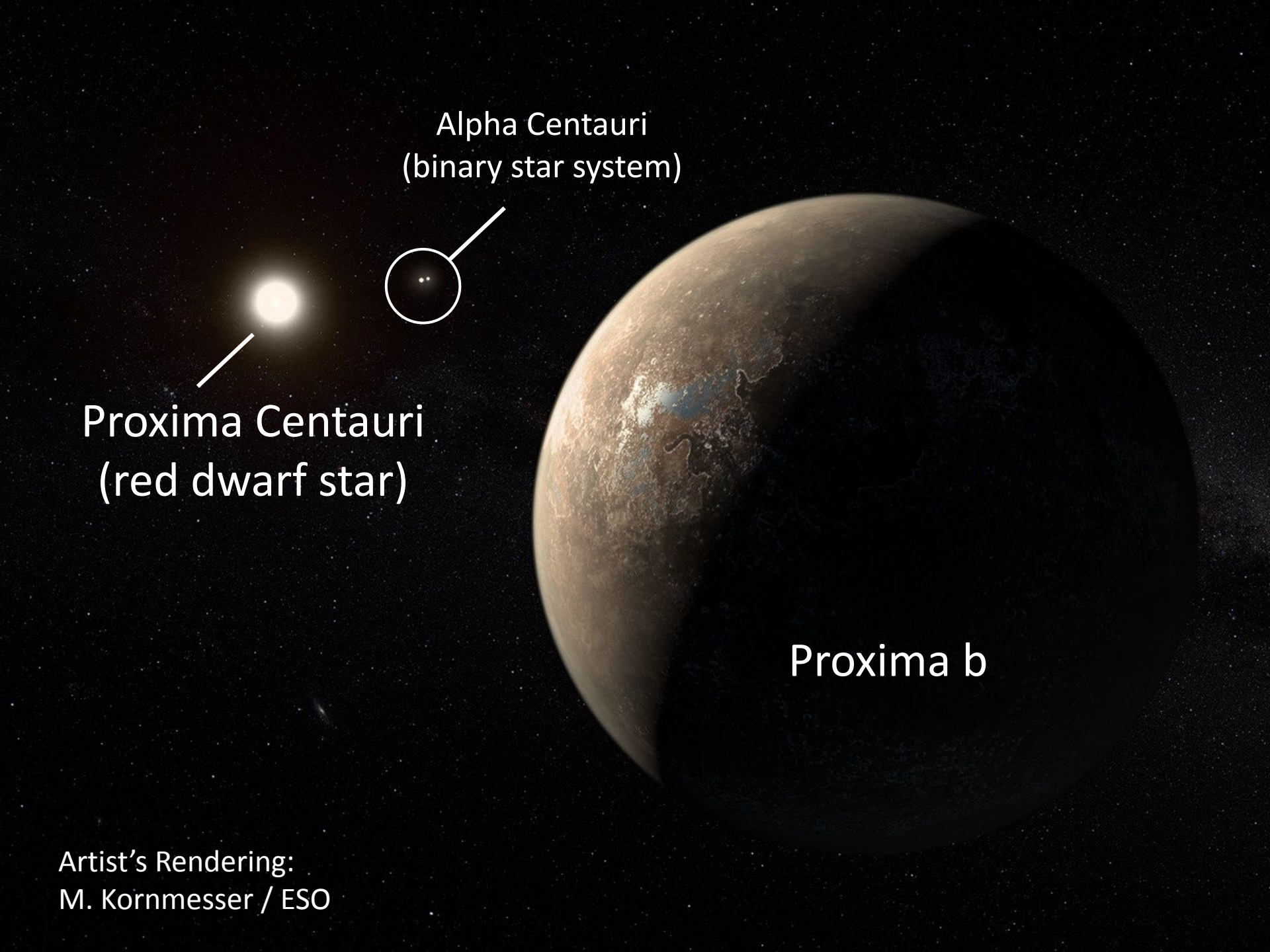
ENGR 101 – Project 1

Introduction and Overview



Artist's Rendering:
M. Kornmesser / ESO

Previously on our show…

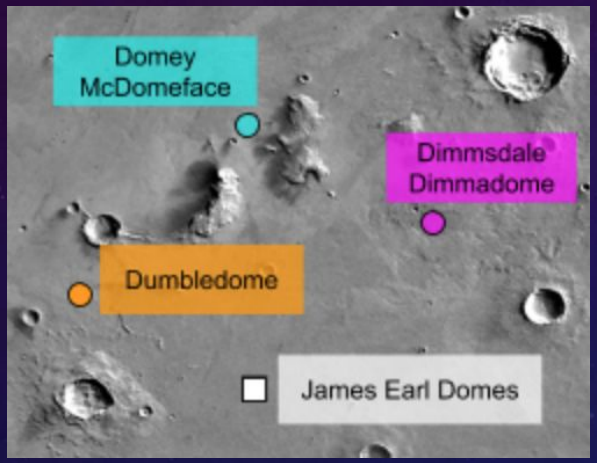
- Sent a rover to analyze soil samples on Proxima b
- Looking for key measures of soil quality for farming
  - pH
  - phosphorus (P)
  - potassium (K)
  - calcium (Ca)
  - magnesium (Mg)
  - sodium (Na)
  - organic matter
  - soil texture, etc.

preliminary results are promising

next step: build settlements!

Timeline: 2 years after the first settlement…

Domed settlements built by previous ENGR 101 students!



The domes provide protection from the elements, but there is a very limited amount of space inside.

Your Job

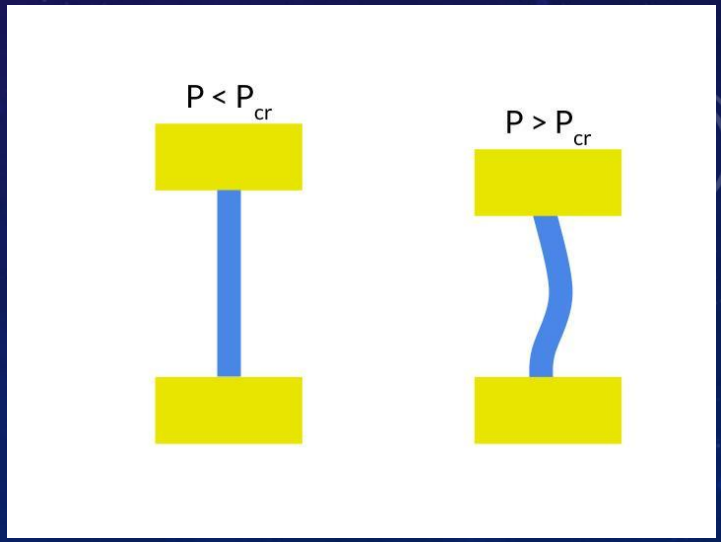
- Write functions to analyze the stability of an underground structure based on the values of several design parameters:
  - **Task 1**: Determine the critical load of support poles.
  - **Task 2**: Determine the load on support poles based on the distribution of weight caused by buildings above the structure.
- Write functions for logistical calculations:
  - **Task 3**: Determine the storage capacity of the structure.
  - **Task 4**: Calculate the total revenue from paid parking in the structure.

Task 1: Critical Load of Support Poles

- Euler's **critical load** is the maximum load a structural column can bear while staying straight.

$$P_{cr} = \frac{\pi^2 EI}{(KL)^2}$$

- *E* - Modulus of elasticity of the column material
- *I* - Minimum area moment of inertia across the cross section of the column
- *K* - Column effective length factor
- *L* - Length of the column



Task 1: The `criticalLoad` Function

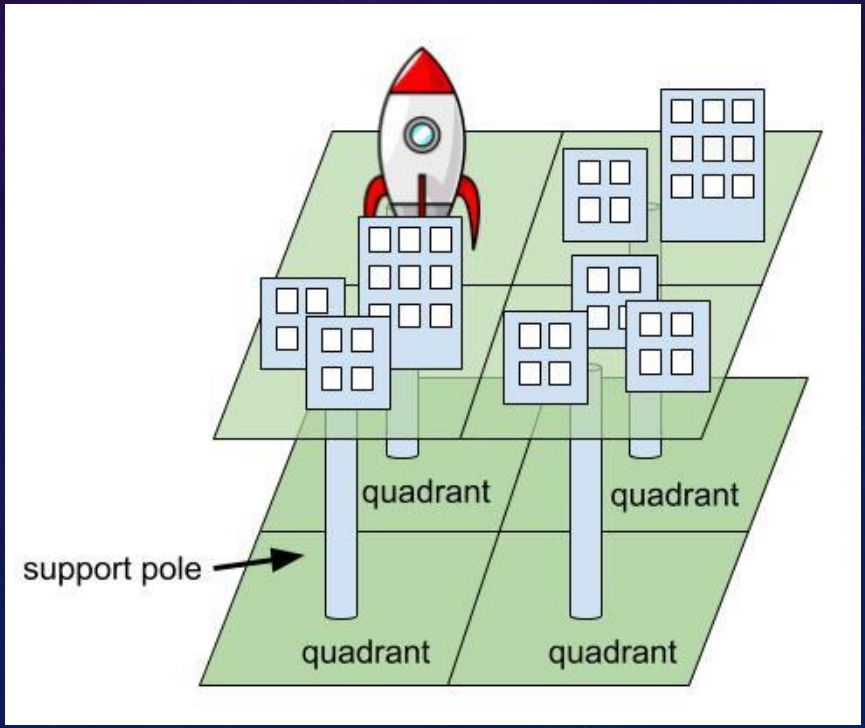
- Write a function to compute the critical load:

```
function [P_cr] = criticalLoad(E, I, K, L)
%criticalLoad computes Euler's critical load for a column with
%  structural parameters: E, I, K, and L.
%     E: Modulus of elasticity of the column material
%     I: Minimum moment of inertia across cross section of the
column
%     K: Column effective length factor
%     L: Length of column
%
%     P_cr: Euler's critical load of column

    % Write your code here!

end
```

- Write your function in the `criticalLoad.m` starter file.
- We've provided a test script in `TestCriticalLoad.m`.

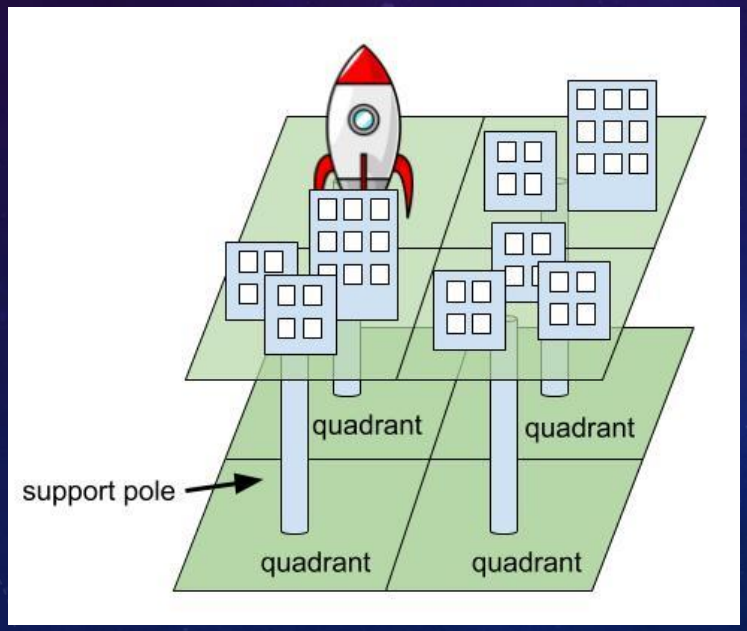
Task 2: Determining the Actual Load

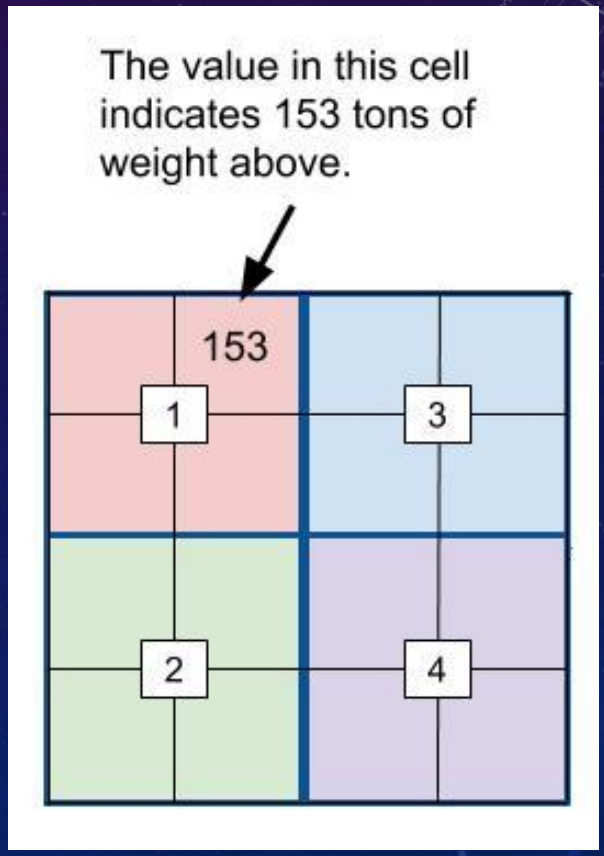
- For this project, we assume the structure is rectangular in shape and that four support poles are used to hold up the "ceiling" (and the buildings on top of it).



Task 2: Determining the Actual Load

- We model the load on each area of the structure using a matrix of weights.
- Each pole will bear the load of weight in its quadrant.



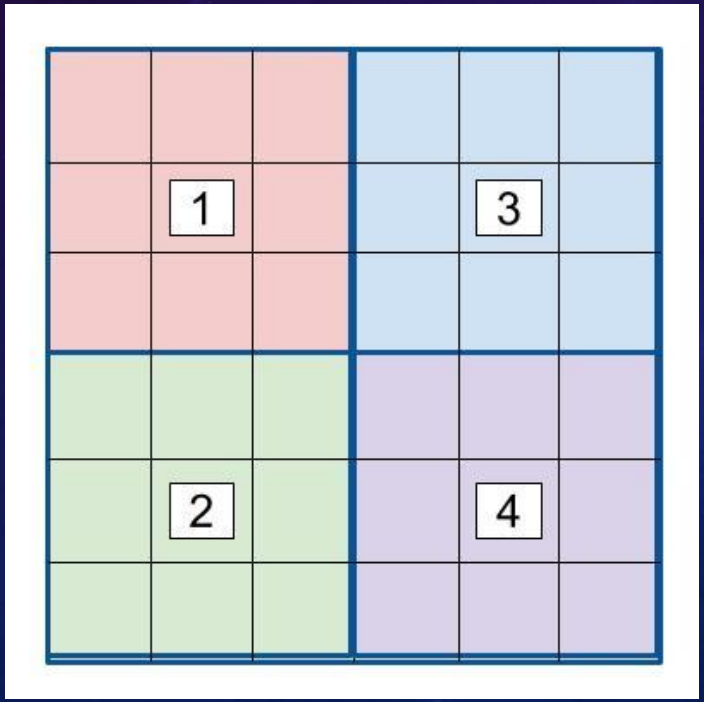


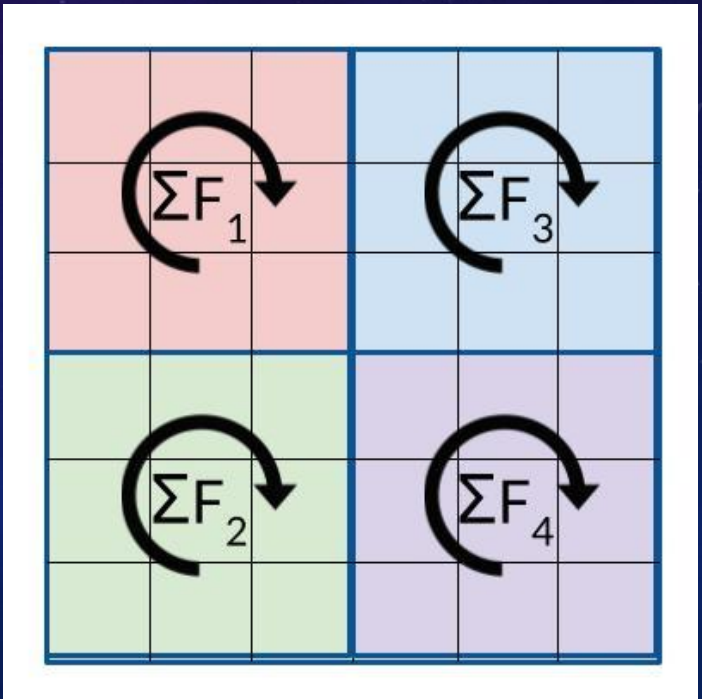
Task 2: Determining the Actual Load

- Question: Which column must bear the largest load?
  - If all columns are identical, this would be the first column to buckle.

Use indexing to select each quadrant.

Find the largest sum of forces in any quadrant.





Task 2: The `actualLoad` Function

- Write a function to compute the largest actual load:

```
function [maxLoad] = actualLoad(W)
    %actualLoad computes the largest load any of the support poles would
    %  need to bear based on the distribution of forces in the W matrix. We
    %  assume each pole is solely responsible for the forces in its
    %  quadrant.
    %     W: Weights matrix (Distribution of weight on the roof)
    %
    %     maxLoad: the largest load among any of the four support poles

    % Write your code here!

end
```

- Write your function in the `actualLoad.m` starter file.
- See Lab 2 for a test script to test your `actualLoad` function!

Task 3: Calculating Storage Space

- Part of the structure is used for long-term storage.
  - Food, medicine, etc. are stored in pallets arranged in a grid.
  - Question: Given the number of pallets already stored, the height of the ceiling, and the height of each pallet… how many more can we fit?

if roof height is 10, and the height of the pallets is 3, then...

1	2	3	2
3	0	2	2
2	1	0	1
1	3	0	2

3 pallets already here. Can't fit any more.

2 pallets already here. Can fit 1 more.

can add 23 total additional pallets

Task 3: The `additionalPallets` Function

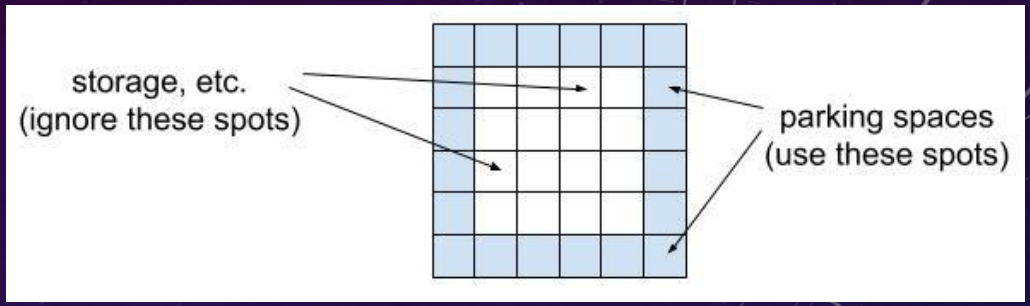
- Write a function to compute how many more pallets we can fit:

```
function [numPallets] = additionalPallets(roofHeight, pallets, palletHeight)
    %additionalPallets computes the number of additional storage pallets
    %     that can fit in the storage area of the structure
    %     roofHeight: scalar representing the height of the roof
    %     pallets: a matrix representing the number of pallets in
    %              each storage cell of the storage area
    %     palletHeight: scalar representing the height of a single pallet
    %
    %     numPallets: number of additional pallets that can fit in
    %                 the storage area

    % Write your code here!

end
```

- Write your function in the `additionalPallets.m` starter file.
- We've provided a test script in `TestAdditionalPallets.m`.

Task 4: Parking Fees (i.e. how to pay for this thing)

- Parking spaces are located on the edge of the structure.



- Goal: Compute the total revenue given:
  - A matrix containing the individual fee for each spot
  - A matrix containing the amount of time the spot was used

0.7	4.2	8.0	6.5	2.5	1.8
0.9	?	?	?	?	5.5
3.3	?	?	?	?	1.1
4.9	?	?	?	?	2.3
6.5	?	?	?	?	7.8
2.2	2.2	3.5	5.2	2.4	0.0

matrix representing the number of hours each space was rented

1.2	1.2	1.2	1.5	1.5	1.5
1.2	?	?	?	?	1.5
0.7	?	?	?	?	0.8
0.7	?	?	?	?	0.8
1.8	?	?	?	?	1.9
1.8	1.8	1.7	1.6	1.9	1.9

matrix representing the price per hour of each space

Task 4: The parkingRevenue Function

- Write a function to compute the total parking revenue:

```
function [revenue] = parkingRevenue(timeUsed, price)
    %parkingRevenue computes the revenue from parking fees
    %     timeUsed: a matrix with the number of hours each spot was used
    %     price: a matrix with the price per hour for each spot
    %     ONLY THE EDGE VALUES FROM EACH MATRIX SHOULD BE USED
    %
    %     revenue: the total amount earned from all parking spots

    % Write your code here!

end
```

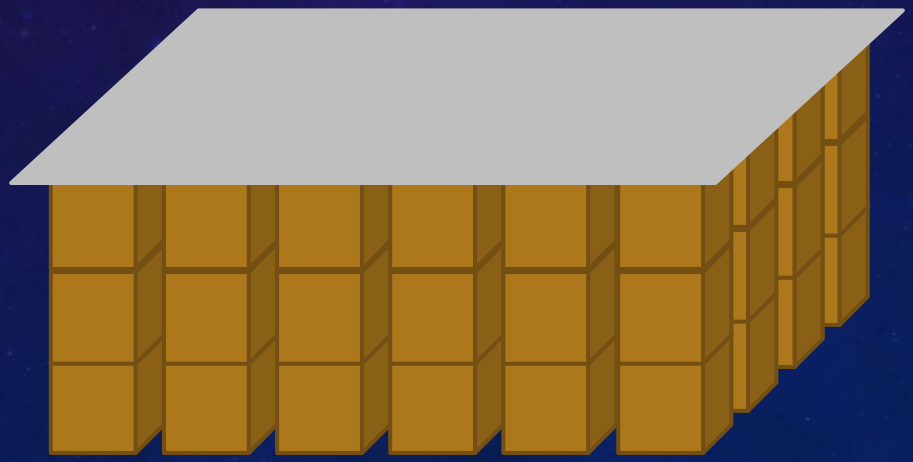
- Write your function in the `parkingRevenue.m` starter file.
- We've provided a test script in `TestParkingRevenue.m`.

Advice: Unit Test Your Functions

- We have provided you with test scripts for each function.
- Each test script defines a set of test data to use with the function -- this is a unit test (see Runestone Ch. 3).
- If your function works correctly, you should get the correct results (see the specs for more detail).
- The test scripts only have one unit test each.
- You should make many more unit tests for each of your functions.

Example: Brainstorming a Unit Test for Task 3

- What if the storage space in Task 3 is already completely filled?
  - What are values of `roofHeight`, `pallets`, and `palletHeight` that describe this situation?
  - What value should be returned by the `additionalPallets` function for this unit test?



Autograder

- You must tell the autograder whether you are working with a partner or alone before you can submit.
  - All project submissions are reviewed for honor code violations!

- Submit these 4 files to the autograder:
  - Task 1 – `criticalLoad.m`
  - Task 2 – `actualLoad.m`
  - Task 3 – `additionalPallets.m`
  - Task 4 – `parkingRevenue.m`

- Limit: 5 submissions/day receive feedback
  - Submissions past the limit are still graded, but no feedback

- See Google Drive > Autograder Guide for details

Autograder - Project Grading

- Correctness: 50 points possible
  - Autograder will run a set of test cases and report back a score
  - Test cases that pass ✅ 👍
  - Test cases that fail will give you some feedback on what was wrong
    - (Submissions after the 5/day limit don't get feedback)
  - Best score kept as your grade

- Style & Comments: 10 points possible
  - Graders will evaluate your script
  - See project specifications for details
  - Latest submission *that is also a best score* will be graded for style & comments