To start you off, I will begin by asking you to create a **relatively** simple array simulation. This may or may not be hard, I am not particularly sure, but I do expect you to ask me questions regarding overall structure and some of the mathematics. I will leave it somewhat ambiguous, because I am not testing your coding ability, but rather your mathematical intuition and problem solving skills with regards to the physical world.

This project will be relatively long, so as long as I see progress happening, you'll be in my good books. The purpose of the array simulation is to give us an idea of how much energy we might be receiving during a race day or over the entire race. This serves two purposes, the first being that we will be able to rank array performances for different designs, the second, which will be prevalent a little later, is that it is data required in order to optimize the race strategy, after all we can't know how to drive if we don't have an idea of how much energy we're getting.

In order to know how the cars array will perform, several things matter to you. The first is the efficiency of the cells, how much solar incident energy can be converted to usable energy by the solar array?

Assume cell efficiency of 25%

Next is how much energy will the sun actually provide us? This is a little bit of a complicated question, irradiance from the sun can come directly from the sun, but it can also come from the scattering of light by the atmosphere. The array is also not technically flat, its composed of tiny prisms on the surface, as such the array efficiency changes with the angle of the sun, it also depends on temperature.

In order to simplify this problem we can assume for now that the efficiency of the array is constant, regardless of the sun's angle. Let's also assume that the atmosphere does not provide a significant amount of light, and we can model all irradiance as coming directly from the sun as rays/vectors. For now **assume the sun to provide 1000W/m^2, never changing** .

Now we have know the efficiency and the incident solar irradiance, but we need to figure out how much energy we might be getting. This is dependent on the angle at which the rays hit the solar cells, the sharper the angle, the less area is "visible" to the sun, and less power is generated. Figuring this out will require some knowledge of linear algebra. This is all fundamentally dependent on how the array has been designed.

Typically, array designs are done with some CAD software, and tend to be complicated, these are typically imported as an .stl type file. We must interpret the file to understand mathematically how the array is shaped. This is quite hard, so the function that does this will be provided to you and explained further on.

We must also consider shading, for simplicity, the only thing we will consider is the shading from the canopy, the area where the driver sits. An stl file will also be provided for the canopy, and any other objects which may cause shading. We have to think about how to accurately model shading caused by non-array components.

To begin with I will give you several functions:

1) `[v, f, n, name] = stlread2(fileName):`
   This function takes in an stl file, and converts it into something readable by matlab. It "meshes" the file, meaning that the CAD is converted into a series of small triangles that describe the surface of the array or canopy, the only output you should concern yourself with is **"v" and "f"**, this outputs the vertices of all of the triangles, and f denotes the indices at which the paired vertices occur.

2) `[Az El] = SolarAzEl(UTC,Lat,Lon,Alt)`
   This function takes in the time (universal), latitude longitude and altitude of your position on earth, and outputs the sun's angle relative to you in terms of azimuth and elevation, azimuth is the degrees from north going clockwise, so an azimuth of 90 is east, and elevation is the angle the sun makes with the ground. This function has a lot of documentation, so check it out, as using it can be tricky sometimes.
3) `vertices_fixed=order_vertices(vertices,faces)`
   This function orders the vertices into something more understandeable, simply use v and f outputed from the stlread2 function as arguments in this function. It will return all the vertices ordered such that each 3 belong to the same triangle, so the first 3 belong to the first triangle, the next 3 to the second etc. column one holds the x coord, 2 the y, and 3 the z, the numbers should be in mm

I will give you more functions, if I have not mentioned them it is simply because they are not important to you, and are used in other functions. In other words, you do need them, but you don't need to touch or use them.

Now the goal of this project is to develop a simulation for the solar array to determine how much energy we might receive in a typical race day, which goes from 8-5. I do not want to tell you how to solve this problem, because I think it will be more fun for you and informative if I let you figure out how to solve the problem. I will however give you hints in the form of what functions you will need, or at least the functions I used when creating the simulation for the first time. You will as such, complete these functions based on their short descriptions, and use them in the final array simulation script. If you get stuck, I will help you, so don't hesitate to ask. The functions range from very easy, to moderately difficult. They are as follow:

Basic Mathematical functions

`area=area_of_triangle(three_by_three)`

takes in 3 points in 3d space, returns the area, simple enough

`total_area=area_of_nsided_convex_polygon(vertices)`

input n number of vertices, and output the area, assuming it is convex (think polar coordinates)

vector=create_sun_vector_simple(azimuth,elevation)

Usually angles for azimuth and elevation are not necessarily useful, try converting these angles into a vector that describes sun rays, make sure that the magnitude of the vector is equal to that of the suns irradiance

binary=point_in_triangle(point,triangle)

This function will take in a point and triangle as arguments, and will determine whether that point exists within a triangle in 3d space. Try to think about how that might be important for shading.

points=project_onto_sun_vector(vertices,sun_vector)
the name of this function is slightly misleading, what it actually wants you to do is to project points onto a plane described by the sun vector. Think about how this might be relevant in determining solar power at different angles.

points=remove_inner_points(unfiltered_points)
this function accepts a set of points which are all assumed to be on some plane, it then requires you to remove all points except for the outermost points that describe the perimeter. A useful function to look at is "convhull/convexhull"

areas=return_areas(vertices)
this function is meant to return the total area of any number of triangles, again, think about how this will be useful

function points_reduced=shading(points,vertices)
this function might requires you to input one set of points, as well as a set of vertices, and see which of the points lie in any of the triangles. It then removes all points that do not exist within any of the triangles. Think how might this be used for shading?

Your job is to complete these functions, as well as decide how to use all of them in an array sim whose goal is to determine solar energy collected in an 8-5 day in a "main" script. I am almost definitely missing some information, so again ask me questions and I will help, the project has been left somewhat vague not to make it difficult but for you to think, since I assume you're already descent at coding. I highly encourage you guys working together to complete this. You also have the freedom to solve it any way you want