For this assignment, I changed the code provided to me to use ray tracing, shadows, reflection, refraction, and depth of field in order to create specific images that matched the images provided in as sample output by implementing these ray tracing effects. For exercise 1, I had to fill the ray_sphere_intersection and ray_parallelogram_intersection functions with the correct code that would alter p and N and return t. For ray_sphere_intersection I had to use sphere_center, sphere_radius, ray_origin, ray_direction, and index in order to accomplish this task. Large calculations were done in order to calculate the necessary value for t and the value for the if statement to compare with 0 to see if it is greater than or equal to with smaller calculations for p and N. For ray_parallelogram_intersection I had to use pgram_origin, A, B, pgram_u, pgram_v, ray_origin, ray_direction, and index in order to accomplish the task. Large calculations were done in order to calculate the necessary boolean value for the if statement and smaller calculations for p and N.

For exercise 2 I was to determine if a point (p) is in the shadow of another or not. I had to cast a ray from the point (p) to the different light sources in the scene. If another object lies in between the point and the light source, then this light does not contribute to the point's color. In order to accomplish this I implemented the diffuse and specular shading with them each needing their own calculation. Next I implemented shadow rays by using the function is_light_visible. The is_light_visible function was made to return a boolean variable. This variable determines whether the light is visible or not. The calculations for this are compared to each other to see if one is greater than the other and the find_nearest_object is used and compared to 0. These comparisons are used in an if statement to determine if the overall is_light_visible function returns true or false. If true is returned we continue;.

For exercise 3 I was to render reflected rays. Using the reflected array and r formulas I had to use recursion with the shoot_ray function while making sure that it wouldn't go on infinitely using max_bounce. For the next part of exercise 3 I was to implement perlin noise, linear interpolation, dotGridGradient function. I had to get the correct grid coordinates from the point x and y for that calculation and replace the linear interpolation with a cubic interpolation formula to get the correct results.

Final Image is below: