Deliverable 3:

Question 1.

The purpose of the algorithm is to generate prime numbers. The function takes a 32-bit unsigned integer n as an input and utilizes Rust's standard library to initiate a random number generator. Within an indefinite loop, it selects a random number within the range of 0 to n, ensuring the number is odd by setting its least significant bit to 1 (this is because even numbers cannot be prime except for 2). The candidate number is converted into a 64-bit unsigned integer for the primality test. The is_prime function checks whether this candidate number is prime. If the candidate is indeed a prime number, the function returns it. Otherwise, the loop persists until a prime number is found.

Question 3.

The purpose of the algorithm is to determine whether a number is likely prime using probability. It begins by representing the candidate number in the form $2^s \times d + 1$. The algorithm then picks a random number within a certain range and uses it to perform a series of checks. If the result of these checks is one of the candidate numbers minus one, the candidate might be prime, and the test continues. If these conditions are not met at any point in the loop, the function immediately concludes that the candidate is not prime. This process repeats several times, each with a new random number, to increase the reliability of the result. If the candidate number consistently passes the checks, the function will likely return true, suggesting that the candidate is a prime number.

Question 4.

glass_pumpkin (https://docs.rs/glass_pumpkin/latest/glass_pumpkin/)