PART - A

Implement the following in Modular Arithmetic:
1. Additive inverse of a number
2. Multiplicative inverse of a number
3. Inverse of an m×m matrix with m ≤ 3
4. Implement Diffie Hellman Key Exchange (DHKE) Protocol, with small integer values for testing. Hence deliver a man-in-the-middle attack on the DHKE Protocol.

PART - B

1. Implement the Euclidean Algorithm below, to find GCD of two numbers:

$$\text{EUCLIDEAN ALGORITHM}(a, b)$$

$$r_0 \leftarrow a$$
$$r_1 \leftarrow b$$
$$m \leftarrow 1$$
$$\textbf{while } r_m \neq 0$$
$$\textbf{do } \begin{cases} q_m \leftarrow \lfloor \frac{r_{m-1}}{r_m} \rfloor \\ r_{m+1} \leftarrow r_{m-1} - q_m r_m \\ m \leftarrow m + 1 \end{cases}$$
$$m \leftarrow m - 1$$
$$\textbf{return } (q_1, \ldots, q_m; r_m)$$
$$\textbf{comment: } r_m = \gcd(a, b)$$

2. Given two integers a and b, the following algorithm computes GCD (a,b) as well as b^{-1} mod a, when a and b are co-prime to each other. This is called the Extended Euclidean Algorithm.

$$\text{EXTENDED EUCLIDEAN ALGORITHM}(a, b)$$

$$a_0 \leftarrow a$$
$$b_0 \leftarrow b$$
$$t_0 \leftarrow 0$$
$$t \leftarrow 1$$
$$s_0 \leftarrow 1$$
$$s \leftarrow 0$$
$$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$$
$$r \leftarrow a_0 - q b_0$$
$$\textbf{while } r > 0$$
$$\textbf{do } \begin{cases} temp \leftarrow t_0 - qt \\ t_0 \leftarrow t \\ t \leftarrow temp \\ temp \leftarrow s_0 - qs \\ s_0 \leftarrow s \\ s \leftarrow temp \\ a_0 \leftarrow b_0 \\ b_0 \leftarrow r \\ q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor \\ r \leftarrow a_0 - q b_0 \end{cases}$$
$$r \leftarrow b_0$$
$$\textbf{return } (r, s, t)$$
$$\textbf{comment: } r = \gcd(a, b) \text{ and } sa + tb = r$$

This is how the above algorithm works:

Given a and b it computes another two number s and t such that s × a + t × b = r = GCD (a,b).

Now, we aim to find b^{-1} mod a, which exists iff a and b are co-prime.

Since a and b are co-prime r = 1.

Therefore, s × a + t × b = 1.

Applying mod a to both sides, (s × a + t × b) mod a = 1 mod a.

Or, t × b mod a = 1 [Since s × a mod a = 0.]

Or, b^{-1} mod a = t. (Think why we also output s.)

Implement the Extended Euclidean Algorithm. Hence prove 28^{-1} mod 75 = 67

<center>**********</center>