Lab 10: Implementation of Diffie–Hellman Key Exchange

Course: Information Security & Cryptography (AI Department, SVNIT Surat)

1. Objective

- Implement the Diffie–Hellman (DH) key exchange protocol in C++.
- Demonstrate how two parties (Alice and Bob) can agree on a shared secret over an insecure channel.
- Show that an eavesdropper cannot compute the shared key even after seeing all public values.

2. Background

The Diffie–Hellman protocol allows two parties to establish a common secret key using modular arithmetic.
Let p be a large prime and g a primitive root modulo p.

1. Alice chooses a secret a, computes $A = g^a \bmod p$, and sends A to Bob.
2. Bob chooses a secret b, computes $B = g^b \bmod p$, and sends B to Alice.
3. Alice computes $K_A = B^a \bmod p$.
4. Bob computes $K_B = A^b \bmod p$.

Then $K_A = K_B = g^{ab} \bmod p$.

Even if an attacker knows (p, g, A, B), recovering the shared key requires solving the **Discrete Logarithm Problem**.

For this lab, use 64-bit primes (for demonstration only, not for real security).

3. Tasks

Task 1: Implement Modular Exponentiation

Write a function:

```
uint64_t modexp(uint64_t base, uint64_t exp, uint64_t mod);
```

Compute $(\text{base}^{\text{exp}}) \bmod \text{mod}$ using repeated squaring.

Task 2: Simulate Alice and Bob

- Fix a prime p and generator g.
- Randomly generate Alice's secret a and Bob's secret b.
- Compute public values $A = g^a \bmod p$, $B = g^b \bmod p$.
- Exchange A and B, then compute shared keys:

$$K_A = B^a \bmod p, \quad K_B = A^b \bmod p.$$

Task 3: Print All Values

Your program should clearly print:

1. p and g
2. Alice's private key a and public value A
3. Bob's private key b and public value B
4. Shared keys K_A and K_B

Sample Output:

```
Public prime p = 2147483647
Generator g   = 5

Alice private a = 173421
Alice public  A = 1189461522

Bob private b = 932884
Bob public  B = 1477719820

Shared key computed by Alice = 681234123
Shared key computed by Bob   = 681234123
Keys match: YES
```

Task 4: Short Report Answer

Explain in 4–6 sentences:

- Why an attacker cannot easily compute the shared key.
- Which mathematical problem they must solve.

4. Submission Guidelines

Submit a single `.zip` file containing:

1. `dh.cpp` — your source code.
2. `output.txt` — one sample program run.

3. `report.pdf` — 2–3 pages including:
    - explanation of DH algorithm,
    - output screenshot,
    - answer to Task 4.

The code must compile with:

```
g++ -std=c++17 dh.cpp -o dh
```

and run without user input.

7. Hints

- Use `uint64_t` for safe 64-bit operations.
- Do not use `pow()`; implement repeated squaring.
- Random keys:

```
std::random_device rd;
std::mt19937_64 gen(rd());
std::uniform_int_distribution<uint64_t> dist(2, p-2);
uint64_t a = dist(gen);
uint64_t b = dist(gen);
```

- Suggested parameters: $p = 4294967311$, $g = 5$.