Lab Exercise 1: Introduction to C++

Submit as .cpp files.

✅ Problem 1: Greeting with Default & Overloaded Functions

Task:
Write two overloaded functions named greet:

1. `greet(string name)` → prints "Hello, [name]!"
2. `greet(string name, string greeting)` → prints "[greeting], [name]!"

Also, write a third version:

3. `greet(string name = "Guest")` → uses a **default argument**

Test in main():

Cpp:

```
greet("Ali");                    // Hello, Ali!
greet("Sara", "Hi");             // Hi, Sara!
greet();                         // Hello, Guest!
```

🎯 ***Focus**: Function overloading + default arguments*

✅ Problem 2: Bitwise Power-of-Two Checker

Task:
Write a function bool isPowerOfTwo(int n) that returns true if n is a power of two (e.g., 1, 2, 4, 8, 16…), else false.

***Hint**: Use bitwise AND (&).*

💡 *Trick: A number n is a power of two **iff** n > 0 and (n & (n - 1)) == 0.*

Sample:

Cpp:

```
isPowerOfTwo(8);  // true
isPowerOfTwo(6);  // false
```

🎨 ***Focus**: Bitwise operators, logical thinking*

✅ Problem 3: Swap Using References

Task:
Write a function void swap(int &a, int &b) that swaps two integers **using references** (not pointers!).

In main(), read two numbers from the user, call swap, and print them.

Sample Input/Output:

```
Enter two numbers: 5 10

After swap: 10 5
```

🎨 ***Focus**: References vs pointers, clean C++ style*

✅ Problem 4: Dynamic Array Sum

Task:
Ask the user for n, then dynamically allocate an array of n integers using new.
Read n numbers into the array, compute their sum, and print it.
Don't forget to delete[] the array!

Sample:

```
How many numbers? 3

Enter numbers: 4 7 2

Sum = 13
```

🎨 ***Focus**: Dynamic memory (new/delete), basic I/O*

✅ Problem 5 (Challenge): Fibonacci with Memoization (Preview of DP)

Task:
Write a recursive function int fib(int n) that computes the nth Fibonacci number **but stores previously computed values in a global array** (size 100, initialized to -1).

This is a **top-down DP (memoization)** intro!

Rules:

- If fib_cache[n] != -1, return it.
- Else, compute it, store it, then return.

Test:

Cpp:

```
cout << fib(10); // Should output 55 quickly, even for n=40
```

🪀 ***Focus**: Recursion + caching (gentle intro to DP concept)*