- Read the problem statement multiple times to identify inputs, outputs, and constraints.
- explain the problem in your own words to ensure clarity. As Einstein reportedly said, “If you can’t explain it simply, you don’t understand it well enough.”

For_each	Syntax :
	for_each(begin, end ,function_name/lambda)
	Ex.
	For array
	for_each(array,array+size [](int x) { cout<<x});
	for Vector
	for_each(vec.begin(),v.end(),[](int x) { cout<<x});
	
	for_each , Lambda take only one argument,
	need two parameters for some operation, std::for_each **isn't** the right tool,

Auto	Deduced at compile time, need to initialized.Function Signatures Must Be Explicit, thats why we can’t pass argument to function, as function signature is important in case function overloading is c++11 design not support this but this support in c++20 as template

Range-Based For Loop	std::vector<int> vec = {1, 2, 3, 4, 5};
	for (const auto& num : vec) { // *Iterates over each element*
	std::cout << num << " ";
	}
	Range-based For Loop with Pointer: Range-based for loops don’t work directly with pointers (int*) because they lack size information. You need an array or a container like std::span (C++20) or a traditional loop.

Lambda Expressions	std::vector<int> vec = {5, 2, 9, 1, 5}; // *Lambda to sort vector in descending order* std::sort(vec.begin(), vec.end(), [](int a, int b) { return a > b; });

Move	Transfer ownership to
	cout<<"\noriginal vector : ";
	vector<int> dest =move(v);
	for (const auto &num:v){
	cout<<" "<< num;
	}

	O/p => empty

Variadic Templates	allow functions or classes to accept a variable number of argument
	Ex. template<typename T, typename... Args>
	void print(T first, Args... args) {
	std::cout << first << " ";
	if constexpr (sizeof...(args) > 0) print(args...); // *Recursive expansion*
	}

Noexcept	Function not throwing any exception.

LValue(locator value)	Rvalue(right hand value)
Has name & persistent storage	Has name but cant persistent storage, temporary storage.
Can take address like (&)	Can't take address
Can be modified(if const not defined)	Can't modified
Ex. Int x, arr[0]	42, x+y
	Denoted &&

shared_ptr, weak_ptr, unique_ptr

unique_ptr

=> Only one unique_ptr can own the object at a time.

Feature : 1. Automatically delete when goes out of scope.
2. Non-Copyble.
3. Move support

Shared_ptr

=>multiple reference count for same object.

Features: 1. Shared ownership, tracks reference count.
2. Deletes the object when the last shared_ptr is destroyed.

weak_ptr

=>std::weak_ptr: Non-owning reference to a shared_ptr-managed object, used to break circular references.

ClassName(const ClassName&) delete=
=>This is c++11 feature which explicitly mark copy constructor as deleted. Means compiler prevent to used them. If tried to copy object , it will give **compile time error.**

Observer pattern , singleton pattern

AWS Services :

- services such as Lambda, API Gateway, DynamoDB, Step Functions, ECS/Fargate, and S3.
- RESTful APIs for internal and external service integration.
  microservices multi-tenant SaaS platform.

DynamoDB

1. key-value and document data models.
2. NO SQL – no relational DB
3. Serverless - don't have to manage the underlying infrastructure.

Lambda:

1. serverless computing service.
2. It allows you to run code without servers.
3. You write functions , then Lambda execute them as HTTP request

API Gateway:

1. It handles API requests
2.

.B.bottlenecks • U_unnecessary work • D Duplicated work(BUD)

0 1 2 3 4 5

1, 6 , 3 , 2, 8, 4

1, 6 , 6 , 2, 8, 4

For I =1 to n {;

temp = 6, temp =arr[i]

J = 2 j = i

While j>0 && arr[j-1] > key

Arr[j+1]= arr[j] .

J =j -1;

}

Arr[j+1]=temp. ,

}

1, 6 , 3 , 2, 8, 4

1.