**Lab Assignment No: 11**

**NAME:** Shubham Takankhar

**ROLLNO:** 54

**SUBJECT CODE**: IT8002

**SUBJECT NAME**: CPP AND JAVA

**GR NO:** 119C0046

**BATCH:** B3

1)Concepts

Concepts are named compile-time predicates which constrain types. They take the following form:

template < template-parameter-list >

concept concept-name = constraint-expression;

where constraint-expression evaluates to a constexpr Boolean.

Constraints should model semantic requirements, such as whether a type is a numeric or hashable.

A compiler error results if a given type does not satisfy

the concept it's bound by (i.e. constraint-expression returns false).

Because constraints are evaluated at compile-time, they can provide more meaningful error messages and runtime safety.

2)Designated initializers

C-style designated initializer syntax.

Any member fields that are not explicitly listed in the designated initializer list are default-initialized.

struct A {

int x;

int y;

int z = 123;

};

A a {.x = 1, .z = 2}; // a.x == 1, a.y == 0, a.z == 2

3)Template syntax for lambdas

Use familiar template syntax in lambda expressions.

auto f = []<typename T>(std::vector<T> v) {

// ...

};

4)Range-based for loop with initializer

This feature simplifies common code patterns, helps keep scopes tight, and offers an elegant solution to a common lifetime problem.

for (std::vector v{1, 2, 3}; auto& e : v) {

std::cout << e;

}

// prints "123"