I. Implicit Conversion (Widening)

1. Determine which assignments are valid without using *casting* and explain why.

Exercise	Code (Valid or Invalid?)	Why? (Implicit or Error?)
1.	`int i = 50; long l = i;`	
2.	`float f = 15.0f; double d = f;`	
3.	`short s = 5; int i = s;`	
4.	`char c = 'Z'; int i = c;`	
5.	`byte b = 10; short s = b;`	
6.	`double d = 25.5; float f = d;`	

II. Explicit Conversion (Casting or Narrowing)

2. Use explicit *casting* to fix compilation errors and predict the potential loss of information.

Instruction: Correct the code in the "Initial Code" column so that it compiles successfully. Then, in the "Prediction" column, indicate what value will be lost or what data will be truncated.

Exercise	Initial Code (Error)	Corrected Code (With Casting)	Result of	Prediction of Information Loss
7.	`double d = 3.14159; int newType = d;`	`int newType = (int) d;`	`System.out.println(newType); 3`	Decimals are lost (`,14159`).
8.	`long l = 1000L; int`			

```
newType =
l;
```

9.

```
int i = 67;
char
newType =
i;
```

10.

```
int i =
300; byte
newType =
i;
```

11.

```
float f =
123.456f;
long
newType =
f;
```

III. Integrated Challenge (Mixed Conversions)

3. Apply the rules of implicit promotion in arithmetic operations and the need for *casting* when assigning the result.

Instruction: Indicate the final data type of the result variable and whether explicit *casting* is needed to store the value.

Exercise	Code	Expected Data Type of the Result (Ex: int, long, double)	Casting Needed (Yes/No)?
12.	`int x = 5; double y = 2.0; double result = x / y;`	`double`	No
13.	`short a = 10; short b = 2; short result = a * b;`		
14.	`long l = 100L; int i = 50; int result = l + i;`		

15.
```
char c = 'a'; int i =
1; int result = c +
i;
```

16.
```
byte b1 = 5; byte b2
= 10; byte result =
b1 + b2;
```