# Part 1 – User-Defined Delegates (5 Problems)

## Problem 1 – Simple Void Delegate

Create a delegate WelcomeMessage that takes a string name and prints a welcome message.  
Call the delegate with your name.  
  
Example Output:  
  
Welcome, Mohamed!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Problem 2 – Delegate with Return Value (Multiple Operations)

Create a delegate MathOperation that takes two integers and returns an integer.  
Write three methods:  
  
Add(int a, int b) → returns a + b  
Subtract(int a, int b) → returns a - b  
Multiply(int a, int b) → returns a \* b  
  
Ask the user for two numbers, then:  
• Use the delegate to call each operation one by one  
• Print all results  
  
Example Output:  
  
Add = 15  
Subtract = 5  
Multiply = 50

## Problem 3 – Delegate Passed to a Method (with Parameters)

Create a delegate DiscountCalculator that takes a double price and returns a double.  
Write a method ShowFinalPrice(double price, DiscountCalculator d) that:  
• Prints the original price  
• Calls the delegate to calculate discounted price  
• Prints the discounted price  
  
Write two discount methods:  
TenPercentOff(double price) → returns price - (10% of price)  
FiftyOff(double price) → returns price - 50  
  
Call ShowFinalPrice twice, once with TenPercentOff and once with FiftyOff.  
  
Example Output:  
  
Original Price: 200  
Final Price: 180  
  
Original Price: 200  
Final Price: 150

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Problem 4 – Multicast Delegate

Create a delegate Notifier with no parameters.  
Write three methods:  
  
SendSMS() → prints "SMS sent"  
SendEmail() → prints "Email sent"  
ShowPopup() → prints "Popup notification shown"  
  
Combine them in a multicast delegate and call it once.  
Expected Output:  
SMS sent

Email sent  
Popup notification shown

## Problem 5 – Lambda Expression with Delegate

Create a delegate NumberCheck that takes an int and returns bool.  
Use a lambda expression to check if a number is positive.  
Test it with -3, 0, and 5 and print:  
  
Expected Output:  
  
-3 is Positive? False  
0 is Positive? False  
5 is Positive? True

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Part 2 – Built-in Delegates (3 Problems)

## Problem 6 – Action

Use an Action<string, int> to print a message multiple times.  
Call it with "Hello" and 3 to print "Hello" three times.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Problem 7 – Func

Use a Func<double, double, double> to calculate:  
• Addition of two numbers  
• Average of two numbers  
  
Call them with 4.5 and 7.5 and print both results.  
  
Example Output:  
  
Add = 12  
Average = 6

## Problem 8 – Predicate

Use a Predicate<string> to check if a string starts with the letter 'A'.  
Test it with {"Apple", "Banana", "Avocado", "Mango"} and print only the matching ones.  
  
Expected Output:  
  
Apple  
Avocado