Exercise 1: Inventory Management System

Product.cs:

```
public class Product
{
 public int ProductId { get; set; }
 public string ProductName { get; set; }
 public int Quantity { get; set; }
 public double Price { get; set; }
 public Product(int id, string name, int quantity, double price)
 {
 ProductId = id;
 ProductName = name;
 Quantity = quantity;
 Price = price;
 }
 public override string ToString()
 {
 return $"ID: {ProductId}, Name: {ProductName}, Qty: {Quantity}, Price: {Price}";
 }
}
```

Inventory.cs

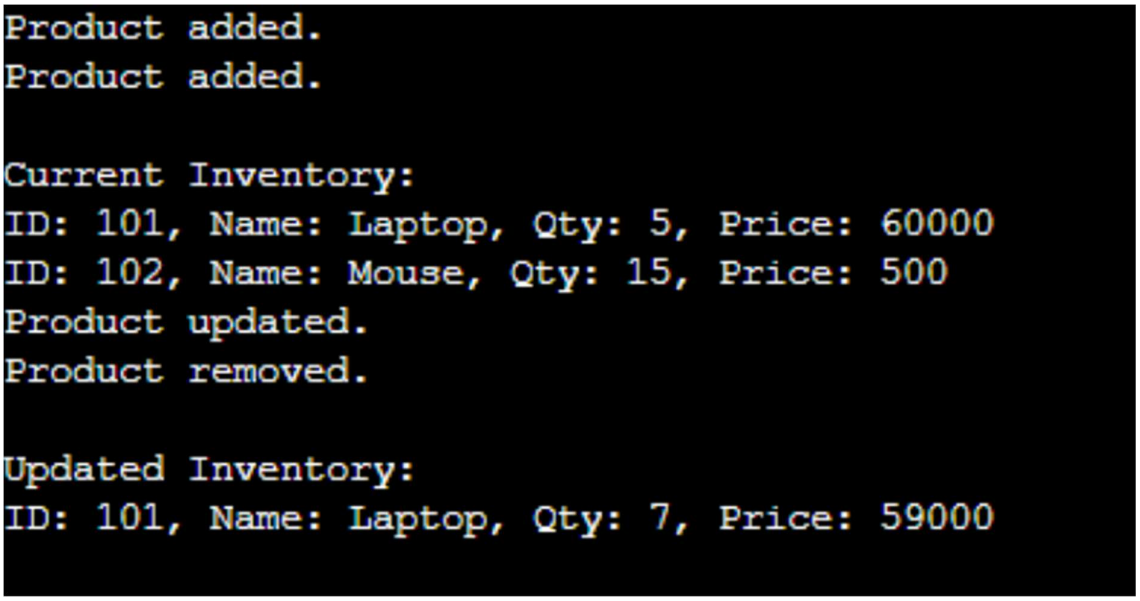
```
using System;
using System.Collections.Generic;
public class Inventory
{
 private Dictionary<int, Product> products = new Dictionary<int, Product>();
 // Add Product
 public void AddProduct(Product product)
 {
 if (!products.ContainsKey(product.ProductId))
 {
```

```
products[product.ProductId] = product;
Console.WriteLine("Product added.");
}
else
{
Console.WriteLine("Product with this ID already exists.");
}
}
// Update Product
public void UpdateProduct(int productId, int quantity, double price)
{
if (products.ContainsKey(productId))
{
products[productId].Quantity = quantity;
products[productId].Price = price;
Console.WriteLine("Product updated.");
}
else
{
Console.WriteLine("Product not found.");
}
}
// Delete Product
public void DeleteProduct(int productId)
{
if (products.Remove(productId))
Console.WriteLine("Product removed.");
else
Console.WriteLine("Product not found.");
}
// Display All Products
```

```
 public void DisplayProducts()
 {
 foreach (var item in products.Values)
 {
 Console.WriteLine(item);
 }
 }
}
```

Program.cs

```
using System;
public class Program
{
 public static void Main(string[] args)
 {
 Inventory inventory = new Inventory();
 // Adding Products
 inventory.AddProduct(new Product(101, "Laptop", 5, 60000));
 inventory.AddProduct(new Product(102, "Mouse", 15, 500));
 // Display Products
 Console.WriteLine("\nCurrent Inventory:");
 inventory.DisplayProducts();
 // Update Product
 inventory.UpdateProduct(101, 7, 59000);
 // Delete Product
 inventory.DeleteProduct(102);
 Console.WriteLine("\nUpdated Inventory:");
 inventory.DisplayProducts();
 }
}
```



Exercise 2: E-commerce Platform Search Function

Product.cs:

```
public class Product
{
 public int ProductId { get; set; }
 public string ProductName { get; set; }
 public string Category { get; set; }
 public Product(int id, string name, string category)
 {
 ProductId = id;
 ProductName = name;
 Category = category;
 }
 public override string ToString()
 {
 return $"ID: {ProductId}, Name: {ProductName}, Category: {Category}";
 }
}
```

Program.cs:

```
using System;
public class Program
{
 public static void Main(string[] args)
 {
 Product[] products = {
 new Product(102, "Laptop", "Electronics"),
 new Product(101, "Shirt", "Clothing"),
 new Product(103, "Book", "Stationery")
 };
 // Linear Search (unsorted array)
 Console.WriteLine("Linear Search:");
 Product result1 = LinearSearch(products, 103);
 Console.WriteLine(result1 != null ? result1.ToString() : "Product not found");
 // Sort the array by ProductId for Binary Search
 Array.Sort(products, (p1, p2) => p1.ProductId.CompareTo(p2.ProductId));
 // Binary Search (sorted array)
 Console.WriteLine("\nBinary Search:");
 Product result2 = BinarySearch(products, 103);
 Console.WriteLine(result2 != null ? result2.ToString() : "Product not found");
 }
 public static Product LinearSearch(Product[] products, int targetId)
 {
 foreach (Product product in products)
 {
 if (product.ProductId == targetId)
 return product;
 }
 return null;
 }
 public static Product BinarySearch(Product[] products, int targetId)
```

```
{
int left = 0, right = products.Length - 1;
while (left <= right)
{
int mid = (left + right) / 2;
if (products[mid].ProductId == targetId)
return products[mid];
else if (products[mid].ProductId < targetId)
left = mid + 1;
else
right = mid - 1;
}
return null;
}
}
```