```
using System;
using System.Collections;

namespace _1_ARRAYLIST
{
    class Program
    {
        static void Main(string[] args)
        {
            #region 01. ArrayList Definition
            /*
             * ArrayList is a non-generic collection in C# that can hold
             * multiple types of data (objects).
             * - Namespace: System.Collections
             * - Stores elements as object type.
             * - Dynamic size (grows/shrinks as needed).
             * - Allows duplicate elements.
             * - Index based access (like arrays).
             */
            #endregion

            #region 02. Initialization of ArrayList
            ArrayList cricketPlayer = new ArrayList();                // Empty ArrayList
            ArrayList arrayList_1 = new ArrayList(cricketPlayer);     // Copy from another
ArrayList
            ArrayList arrayList_2 = new ArrayList(50);                // Initialize with capacity
50
            #endregion

            #region 03. Adding Elements
            cricketPlayer.Add("Rohit");               // Add string
            cricketPlayer.Add(10);                    // Add int
            cricketPlayer.Add("Sachin");              // Add another string
            cricketPlayer.AddRange(arrayList_1);      // Add all elements from another ArrayList
            #endregion

            #region 04. Access Elements
            Console.WriteLine("First element: " + cricketPlayer[0]); // Direct index access
            #endregion

            #region 05. Looping through ArrayList
            Console.WriteLine("\nUsing for loop:");
            for (int i = 0; i < cricketPlayer.Count; i++)
            {
                Console.WriteLine(cricketPlayer[i]);
            }

            Console.WriteLine("\nUsing foreach loop:");
            foreach (var player in cricketPlayer)
            {
                Console.WriteLine(player);
            }
            #endregion

            #region 06. Clear Elements
            arrayList_1.Clear();
            Console.WriteLine("\nCount of ArrayList 1 after Clear(): " + arrayList_1.Count);
            #endregion

            #region 07. Check Existence
            Console.WriteLine("\nContains 10? " + cricketPlayer.Contains(10)); // True/False
            #endregion
```

```
            #region 08. Insert Elements
            cricketPlayer.Insert(0, 45);                        // Insert at index 0
            cricketPlayer.InsertRange(1, new ArrayList()        // Insert range after index 0
            {
                "Virat", "Dhoni"
            });
            #endregion

            #region 09. Remove Elements
            /*
             * Remove(value)     → Removes first occurrence of the given value
             * RemoveAt(index)   → Removes element at given index
             * RemoveRange(start, count) → Removes multiple elements
             */

            Console.WriteLine("\nBefore Removing:");
            foreach (var item in cricketPlayer)
                Console.Write(item + " ");

            cricketPlayer.Remove(10);         // Removes value "10"
            cricketPlayer.RemoveAt(0);        // Removes element at index 0
            if (cricketPlayer.Count >= 2)
                cricketPlayer.RemoveRange(0, 2); // Removes 2 elements starting at index 0

            Console.WriteLine("\n\nAfter Removing:");
            foreach (var item in cricketPlayer)
                Console.Write(item + " ");
            Console.WriteLine();
            #endregion

            #region 10. Clone ArrayList
            ArrayList arrayList_3 = (ArrayList)cricketPlayer.Clone(); // Creates a shallow copy
            Console.WriteLine("\nCloned ArrayList count: " + arrayList_3.Count);
            #endregion

            #region 11. Sort ArrayList
            /*
             * Sorting works only if all elements are of the same type.
             * Otherwise, InvalidOperationException will be thrown.
             * So here, we'll create a new ArrayList with same-type elements.
             */
            ArrayList numbers = new ArrayList() { 5, 2, 8, 1, 3 };
            numbers.Sort();
            Console.WriteLine("\nSorted Numbers:");
            foreach (var num in numbers)
            {
                Console.Write(num + " ");
            }
            Console.WriteLine();
            #endregion

            #region 12. CopyTo Example
            object[] array = new object[cricketPlayer.Count];
            cricketPlayer.CopyTo(array, 0);
            Console.WriteLine("\nCopied Array Elements:");
            foreach (var item in array)
            {
                Console.Write(item + " ");
            }
            #endregion
        }
    }
}
```

```
using System;
using System.Collections;

namespace _02_HashTable
{
    class Program
    {
        static void Main(string[] args)
        {
            #region 01. Hashtable Definition
            /*
             * Hashtable (Non-Generic Collection)
             * - Namespace: System.Collections
             * - Stores data as Key-Value pairs (object type).
             * - Key must be unique, Value can be duplicate.
             * - Unordered collection → items are not stored in insertion order.
             * - Allows different data types for Key & Value (since object type).
             * - Similar to Dictionary<TKey, TValue> but non-generic.
             */
            #endregion

            #region 02. Initialization of Hashtable
            Hashtable Country = new Hashtable(); // Empty Hashtable
            #endregion

            #region 03. Adding Elements
            Country.Add("IND", "India");                        // Add key-value
            Country.Add("USA", "United States of America");
            Country.Add("PAK", "Pakistan");
            Country.Add("AUS", "Australia");
            Country["ENG"] = "England";                         // Another way to insert/update
            #endregion

            #region 04. Count of Elements
            Console.WriteLine("Total Countries: " + Country.Count);
            #endregion

            #region 05. Existence Check
            Console.WriteLine("\nCheck existence:");
            Console.WriteLine("Contains 'IND'? " + Country.Contains("IND"));          // True
            Console.WriteLine("ContainsKey 'AUS'? " + Country.ContainsKey("AUS"));    // True
            Console.WriteLine("ContainsValue 'India'? " + Country.ContainsValue("India")); // True
            #endregion

            #region 06. Remove Elements
            Country.Remove("USA"); // Remove by Key
            Console.WriteLine("\nAfter removing 'USA', total count: " + Country.Count);
            #endregion

            #region 07. Access Elements by Key
            Console.WriteLine("\nAccess element by key 'IND': " + Country["IND"]);
            #endregion

            #region 08. Looping Through Hashtable
            Console.WriteLine("\nLoop using foreach (DictionaryEntry):");
            foreach (DictionaryEntry entry in Country)
            {
                Console.WriteLine("Key = " + entry.Key + ", Value = " + entry.Value);
            }

            Console.WriteLine("\nLoop through Keys:");
            foreach (var key in Country.Keys)
```

```
            {
                Console.WriteLine("Key = " + key + ", Value = " + Country[key]);
            }

            Console.WriteLine("\nLoop through Values:");
            foreach (var val in Country.Values)
            {
                Console.WriteLine("Value = " + val);
            }
            #endregion

            #region 09. Clone Hashtable
            Hashtable IPL = (Hashtable)Country.Clone(); // Shallow copy
            Console.WriteLine("\nCloned Hashtable count: " + IPL.Count);
            #endregion

            #region 10. CopyTo Example
            object[] keys = new object[Country.Count];
            object[] values = new object[Country.Count];

            Country.Keys.CopyTo(keys, 0);
            Country.Values.CopyTo(values, 0);

            Console.WriteLine("\nCopied Keys:");
            foreach (var key in keys)
                Console.Write(key + " ");

            Console.WriteLine("\nCopied Values:");
            foreach (var val in values)
                Console.Write(val + " ");
            Console.WriteLine();
            #endregion
        }
    }
}
```

```
using System;
using System.Collections;

namespace StackCollectionDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            #region 01. Stack Definition
            /*
             * Stack (Non-Generic Collection)
             * - Namespace: System.Collections
             * - Stores objects in LIFO (Last In, First Out) order.
             * - Methods:
             *   Push(object)  → Add item to top of stack
             *   Pop()         → Remove + return top item
             *   Peek()        → Return top item without removing
             *   Contains(obj) → Check if an item exists
             *   Clear()       → Remove all items
             *   Clone()       → Create shallow copy
             * - Similar to generic Stack<T> but stores object type.
             */
            #endregion

            #region 02. Initialization
            Stack history = new Stack(); // Empty Stack
            #endregion
```

```
            #region 03. Adding Items (Push)
            history.Push("www.google.com");
            history.Push("www.amazon.in");
            history.Push("www.flipkart.in");
            history.Push("www.grok.ae");
            #endregion

            #region 04. Count of Items
            Console.WriteLine("Total items in stack: " + history.Count);
            #endregion

            #region 05. Printing Stack Elements
            Console.WriteLine("\nStack Elements (Top to Bottom):");
            foreach (object item in history)
            {
                Console.WriteLine(item);
            }
            #endregion

            #region 06. Remove Elements (Pop)
            Console.WriteLine($"\nDeleted Element (Pop): {history.Pop()}"); // Removes top element
            Console.WriteLine($"Stack Count after Deletion: {history.Count}");
            #endregion

            #region 07. Peek (View Top Element)
            Console.WriteLine($"Topmost Element (Peek): {history.Peek()}");
            #endregion

            #region 08. Contains
            Console.WriteLine("\nCheck Contains 'www.google.com': " +
history.Contains("www.google.com"));
            #endregion

            #region 09. Clone
            Stack webHistory = (Stack)history.Clone(); // Shallow copy
            Console.WriteLine("\nCloned Stack Count: " + webHistory.Count);
            #endregion

            #region 10. Clear
            history.Clear();
            Console.WriteLine("Original Stack Count after Clear(): " + history.Count);
            #endregion

            #region 11. CopyTo Example
            object[] objectHistory = new object[webHistory.Count];
            webHistory.CopyTo(objectHistory, 0);

            Console.WriteLine("\nCopied Elements from Cloned Stack:");
            foreach (var item in objectHistory)
                Console.WriteLine(item);
            #endregion
        }
    }
}
```

```
using System;
using System.Collections;

namespace _4_Queue
{
    class Program
    {
```

```
        static void Main(string[] args)
        {
            #region 01. Queue Definition
            /*
             * Queue (Non-Generic Collection)
             * - Namespace: System.Collections
             * - Stores objects in FIFO (First In, First Out) order.
             * - Methods:
             *   Enqueue(object) → Add item at the back
             *   Dequeue()       → Remove + return front item
             *   Peek()          → Return front item without removing
             *   Contains(obj)   → Check if item exists
             *   Clear()         → Remove all items
             *   Clone()         → Create shallow copy
             * - Similar to generic Queue<T>, but stores object type.
             */
            #endregion

            #region 02. Initialization
            Queue line = new Queue(); // Empty queue
            #endregion

            #region 03. Adding Items (Enqueue)
            line.Enqueue("First Person");
            line.Enqueue("Second Person");
            line.Enqueue("Third Person");
            #endregion

            #region 04. Remove Item (Dequeue)
            Console.WriteLine("Removed (Dequeue): " + line.Dequeue()); // Removes first item
            #endregion

            #region 05. Peek (Front Element)
            Console.WriteLine("Front Element (Peek): " + line.Peek()); // Shows next item without
removing
            #endregion

            #region 06. Count
            Console.WriteLine("Queue Count: " + line.Count);
            #endregion

            #region 07. Printing Queue Elements
            Console.WriteLine("\nQueue Items (Front to Back):");
            foreach (var item in line)
            {
                Console.WriteLine(item);
            }
            #endregion

            #region 08. Contains
            bool isValid = line.Contains("First Person");
            Console.WriteLine("\nContains 'First Person'? " + isValid);
            #endregion

            #region 09. Clone
            Queue line2 = (Queue)line.Clone(); // Creates shallow copy
            Console.WriteLine("Cloned Queue Count: " + line2.Count);
            #endregion

            #region 10. Clear
            line.Clear();
            Console.WriteLine("Original Queue Count after Clear(): " + line.Count);
            #endregion
```

```
            #region 11. CopyTo Example
            object[] line3 = new object[line2.Count];
            line2.CopyTo(line3, 0);

            Console.WriteLine("\nCopied Elements from Cloned Queue:");
            foreach (var item in line3)
                Console.WriteLine(item);
            #endregion
        }
    }
}
```

```
using System;
using System.Collections;

namespace _5_SortedList
{
    class Program
    {
        static void Main(string[] args)
        {
            #region 01. SortedList Definition
            /*
             * SortedList (Non-Generic Collection)
             * - Namespace: System.Collections
             * - Stores data as Key-Value pairs.
             * - Automatically sorts elements by Key (ascending order).
             * - Key must be unique, Value can be duplicate.
             * - Allows access by:
             *   - Key (like Hashtable/Dictionary)
             *   - Index (like ArrayList)
             * - Similar to Dictionary<TKey, TValue> but keeps sorted order.
             */
            #endregion

            #region 02. Initialization
            SortedList numbers = new SortedList();
            #endregion

            #region 03. Adding Elements
            numbers.Add(10, "Ten");
            numbers.Add(1, "One");
            numbers.Add(2, "Two");
            numbers.Add(0, "Zero"); // Will be placed first since SortedList auto-sorts by Key
            #endregion

            #region 04. Access Elements
            Console.WriteLine("Access by Key [0]: " + numbers[0]);              // Access using
key
            Console.WriteLine("Access by Index [2]: " + numbers.GetByIndex(2)); // Access using
index
            #endregion

            #region 05. Looping Elements
            Console.WriteLine("\nUsing For Loop:");
            for (int i = 0; i < numbers.Count; i++)
            {
                Console.WriteLine($"Key: {numbers.GetKey(i)}, Value: {numbers.GetByIndex(i)}");
            }

            Console.WriteLine("\nUsing foreach (DictionaryEntry):");
            foreach (DictionaryEntry item in numbers)
            {
```

```
                Console.WriteLine($"Key: {item.Key}, Value: {item.Value}");
            }
            #endregion

            #region 06. Remove Elements
            numbers.Remove(10);   // Remove by Key
            numbers.RemoveAt(2);  // Remove by Index
            Console.WriteLine("\nAfter Removal, Count = " + numbers.Count);
            #endregion

            #region 07. Contains (Search)
            Console.WriteLine("\nContains Value 'One'? " + numbers.Contains("One"));        //
True
            Console.WriteLine("Contains Key 1? " + numbers.ContainsKey(1));                 //
True
            Console.WriteLine("Contains Value 'Two'? " + numbers.ContainsValue("Two"));     //
True
            #endregion

            #region 08. Clone
            SortedList list = (SortedList)numbers.Clone(); // Shallow copy
            Console.WriteLine("\nCloned SortedList Count: " + list.Count);
            #endregion

            #region 09. Clear
            numbers.Clear(); // Clears all elements
            Console.WriteLine("Original SortedList Count after Clear(): " + numbers.Count);
            #endregion

            #region 10. CopyTo Example
            object[] objKey = new object[list.Count];
            object[] objValue = new object[list.Count];

            list.Keys.CopyTo(objKey, 0);
            list.Values.CopyTo(objValue, 0);

            Console.WriteLine("\nCopied Keys:");
            foreach (var key in objKey)
                Console.Write(key + " ");

            Console.WriteLine("\nCopied Values:");
            foreach (var val in objValue)
                Console.Write(val + " ");
            Console.WriteLine();
            #endregion
        }
    }
}
```