|  |
| --- |
| 31st Morning Assignment  By Surya Teja Chandolu |

|  |
| --- |
| 1. Create a simple program to declare ArrayList and assign some values and find sum. |
| Code: |
| using System;  using System.Collections;  namespace SumUsingArrayList  {  internal class Program  {  static void Main(string[] args)  {  int sum = 0;  ArrayList data = new ArrayList();  data.Add(10);  data.Add(20);  data.Add(30);  data.Add(40);  data.Add(50);  foreach(var item in data)  {  sum = sum + Convert.ToInt32(item);  }  Console.WriteLine($"Sum of data is {sum}");  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| 1. Research and find how the values of ArrayList are stored in the memory. |
| A System.Collections.ArrayList object is a sophisticated version of an array. The ArrayList class provide some features that are offered in most System.Collections classes but that are not in the Array class. For example:  The capacity of an Array is fixed, whereas the capacity of an ArrayList is automatically expanded as required. If the capacity of an ArrayList is changed, the memory reallocation and copying of elements are automatically done.  ArrayList provide methods that add, insert, or remove a range of elements. In Array, you can get or set the value of only one element at a time.  ArrayList is a Reference Type, but not Typesafe and less efficient  ArrayList is array-based structures. Therefore they will have a size something like: Capacity\*sizeof(T). Since ArrayList stores things internally in an object[] it will have an additional pointer reference for each item. |

|  |
| --- |
| 1. What are the dis-advantages of ArrayList (Collections ArrayList) |
| * ArrayList is that it holds only object types. * We need to un-box the variables every time. * ArrayList has more Run time error. |

|  |
| --- |
| 1. Create a simple program to declare List<int> and assign some values and find sum |
| Code: |
| using System;  using System.Collections.Generic;  namespace SumUsingList  {  internal class Program  {  static void Main(string[] args)  {  int sum = 0;  List<int> data = new List<int>();  data.Add(10);  data.Add(20);  data.Add(30);  data.Add(40);  data.Add(50);    foreach (int l in data)  {  sum = sum + l;  }  Console.WriteLine($"Sum of values in List is {sum}");  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| 1. In a tabular format write the differences between Collections and generics. |
| |  |  |  | | --- | --- | --- | |  | **Collection** | **Generic** | | **Name space** | System.Collections.ArrayList | System.Collections.Generic.List | | **Element type** | Object | Primitive data type(Ex: int, string,…) | | **Type Casting** | Yes | No | | **Example** | ArrayList data = new ArrayList(); | List<int> data = new List<int>(); | |

|  |
| --- |
| 1. Research and find how the values of List<T> are stored in the memory. |
| A System.Collections.Generic.List<T> object is a sophisticated version of an array. The List<T> generic class provide some features that are offered in most System.Collections classes but that are not in the Array class. For example:  The capacity of an Array is fixed, whereas the capacity of a List<T> is automatically expanded as required. If the capacity of a List<T> is changed, the memory reallocation and copying of elements are automatically done.  List<T> provide methods that add, insert, or remove a range of elements. In Array, you can get or set the value of only one element at a time.  List<T> or Generic list is a Reference Type,but is Type Safe and efficient  List<T> is array-based structures. Therefore they will have a size something like: Capacity\*sizeof(T). Since ArrayList stores things internally in an object[] it will have an additional pointer reference for each item (if the T is a value type). |

|  |
| --- |
| 1. WACP to declare List<String> and add 5 values and print the values using   a. for loop  b. foreach loop  c. Lambda Expression |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace PrintListOfString  {  internal class Program  {  static void Main(string[] args)  {  int size;  List<string> data = new List<string>();  Console.Write("Enter list size: ");  size = Convert.ToInt32(Console.ReadLine());  for (int i = 0; i < size; i++)  {  Console.Write($"Enter {i} element: ");  data.Add(Console.ReadLine());  }  Console.WriteLine("For Loop");  for (int i = 0; i < data.Count; i++)  Console.WriteLine(data[i]);  Console.WriteLine("\nForEach Loop");  foreach (string s in data)  Console.WriteLine(s);  Console.WriteLine("\nLambda Expression");  data.ForEach(p => Console.WriteLine(p));  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| 1. WACP to declare List<int> and read 5 values from user and find sum using   a. for loop  b. foreach loop  c. Lamdba Expression |
| Code: |
| using System;  using System.Collections.Generic;  namespace SumOfListInt  {  internal class Program  {  static void Main(string[] args)  {  int size, num, sum = 0, sum1 = 0, sum2 = 0;  List<int> data = new List<int>();  Console.Write("Enter list size: ");  size = Convert.ToInt32(Console.ReadLine());  for (int i = 0; i < size; i++)  {  Console.Write($"Enter {i} element: ");  data.Add(Convert.ToInt32(Console.ReadLine()));  }    //For Loop  for (int i = 0; i < data.Count; i++)  sum = sum + data[i];  //ForEach Loop  foreach (int n in data)  sum1 = sum1 + n;  //Lambda Expression  data.ForEach(x => sum2 = sum2 + x);  Console.WriteLine("\nFor Loop");  Console.WriteLine(sum);  Console.WriteLine("\nForEach Loop");  Console.WriteLine(sum1);  Console.WriteLine("\nLambda Expression");  Console.WriteLine(sum2);  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| 1. In a tabular format write all data types in C# and write the respective alias name |
| |  |  | | --- | --- | | Data Type | Alias Name | | byte | Byte | | ushort | UInt16 | | uint | UInt32 | | ulong | UInt64 | | sbyte | sByte | | short | Int16 | | int | Int32 | | long | Int64 | | float | Single | | double | Double | | decimal | Decimal | | bool | Boolean | | char | Char | | string | String | |

|  |
| --- |
| 1. Write example programs for implicit and explicit type casting. |
| Code: |
| using System;  namespace ImplicitAndExplicit  {  internal class Program  {  static void Main(string[] args)  {  //Implicit type casting  short shortValue = 10;  int intValue = shortValue;  Console.WriteLine($"Short Value is {shortValue} and Integer Value is {intValue}.");  //Explicit type casting  double doubleValue = 5.7;  float floatValue = Convert.ToSingle(doubleValue);  Console.WriteLine($"Double Value is {doubleValue} and Float Value is {floatValue}.");  Console.ReadLine();  }  }  } |
| Output: |
|  |