| **DAY 6 : Morning Assignment**  **By**  **Vihar D.** |
| --- |

| **Assignment 1** |
| --- |
| **Write a C# program to create an array list with 5 items and find its sum** |
| **Code :** |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace arraylist\_sum  {  internal class Program  {  static void Main(string[] args)  {  ArrayList data = new ArrayList();  int sum = 0;  data.Add(5);  data.Add(10);  data.Add(15);  data.Add(20);  data.Add(25);  foreach (var d in data)  {  sum = sum + (int)d;  }  Console.WriteLine("The Sum of Array List items is : {0}", sum);  Console.ReadLine();  }  }  } |
| **Output :** |
|  |

| **Assignment 2** |
| --- |
| **Research and find how the values of an arraylist are stored in the memory** |
| **Answer :** |
| ArrayList changes memory allocation as it grows.   * When a certain capacity is specified while initializing the ArrayList, the space is allocated to store objects up to that given capacity. Although the logical size of the ArrayList remains 0. * When it is required that the capacity needs to be expanded then the larger array is created and the values are copied to it. * The internal array of ArrayList is of object type, so every value type is essentially boxed and stored on the heap and every element of the ArrayList is a reference to a boxed value type. * When the value type element is accessed, it is unboxed before being able to use it. |
| **Output :** |
|  |

| **Assignment 3** |
| --- |
| **What are the dis-advantages of arraylist (Collections ArrayList) ?** |
| **Answer :** |
| The one most possible disadvantage of the ArrayList is that it only holds object types and none of the other primitive types (eg. int).   * If a data entry is added to or removed from an ArrayList, data needs to be shifted to update the list. * Items in the Arraylist need to be unboxed everytime. * It is definitely hard to maintain uniqueness of the |
| **EXAMPLE :** |
| **namespace** *CollectionDemo*  **{**  **class** Program  **{**  **static** **void** Main**(**string**[]** args**)**  **{**  ArrayList Numbers = new ArrayList**(**3**)**;  Numbers.Add**(**100**)**;  Numbers.Add**(**200**)**;  Numbers.Add**(**300**)**;  Numbers.Add**(**400**)**;  **foreach(int** Number in Numbers**)**  **{**  Console.Write**(**Number + " "**)**;  **}**  Console.ReadKey**()**;  **}**  **}**  **}** |

| **Assignment 4** |
| --- |
| **Write a C# program to declare List<int> and assign some values and find its sum** |
| **Code :** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace listint\_findsum  {  internal class Program  {  static void Main(string[] args)  {  List<int> data = new List<int>();  int sum = 0;  data.Add(5);  data.Add(10);  data.Add(15);  data.Add(20);  data.Add(25);  foreach (int i in data)  {  sum = sum + i;  }  Console.WriteLine("The sum Of the items is : {0}", sum);  Console.ReadLine();  }  }  } |
| **Output :** |
|  |

| **Assignment 5** |
| --- |
| **Write the difference between Collections and generics** |
| **Answer :** |
| | **Collections** | **Generic** | | --- | --- | | 1. **namespace :**   System.Collections | 1. **namespace :**   System.Collections.Generic | | 1. Every element is of object data type in collections. | 2. Every element is of a specified datatype while initializing. | | 1. Type casting is required in collections. | 3. Type casting is not required in Generics. | | 1. **Syntax :**   ArrayList data = new ArrayList(); | **4.** **Syntax :**  List<type> data = new List<type>();  // ( <type> => datatype ) | | 1. **Example**   *// C# to illustrate the concept*  *// of non-generic collection using Queue*  using System;  using System.Collections;    class GFG {    *// Driver code*  public static void Main()  {    *// Creating a Queue*  Queue myQueue = new Queue();    *// Inserting the elements into the Queue*  myQueue.Enqueue("C#");  myQueue.Enqueue("PHP");  myQueue.Enqueue("Perl");  myQueue.Enqueue("Java");  myQueue.Enqueue("C");    *// Displaying the count of elements*  *// contained in the Queue*  Console.Write("Total number of elements present in the Queue are: ");    Console.WriteLine(myQueue.Count);    *// Displaying the beginning element of Queue*  Console.WriteLine("Beginning Item is: " + myQueue.Peek());  }  } | **5.** **Example**  *// C# program to illustrate the concept*  *// of generic collection using List<T>*  using System;  using System.Collections.Generic;    class Geeks {    *// Main Method*  public static void Main(String[] args)  {    *// Creating a List of integers*  List<int> mylist = new List<int>();    *// adding items in mylist*  for (int j = 5; j < 10; j++) {  mylist.Add(j \* 3);  }    *// Displaying items of mylist*  *// by using foreach loop*  foreach(int items in mylist)  {  Console.WriteLine(items);  }  }  } | |
| **Output :** |
| |  |  | | --- | --- | |

| **Assignment 6** |
| --- |
| **Research and find how the values of List<T> are stored in the memory ?** |
| **Answer :** |
| * In a List<T>, the memory to the store values is within the memory allocated for the system. * Lists are stored in distinct chunks of memory which are linked together with pointers which enables efficient use of memory generally and does not require resizing. * It also allows for easy and quick manipulation of pointers when transforming the list. * It is non-contiguous memory and also a non-index based structure. |
| **Output :** |
|  |

| **Assignment 7** |
| --- |
| **Write a C# program to declare a List<string> with 5 values from user and print using for, foreach & lambda expression loops types .** |
| **Code :** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace liststring3looptypes  {  internal class Program  {  static void Main(string[] args)  {  List<string> data = new List<string>();  data.Add("Vihar");  data.Add("Manoj");  data.Add("Vamsi");  data.Add("Pavan");  data.Add("Sharath");  *//Using For Loop------------------------------------------*  Console.WriteLine("\n Output ( Using For Loop ) : ");  for (int i = 0; i < data.Count; i++)  {  Console.WriteLine("\t- {0} -", data[i]);  }    *//Using For Each Loop--------------------------------------*  Console.WriteLine("\n Output ( Using For Each Loop ) : ");  foreach (var d in data)  {  Console.WriteLine("\t- {0} -", d);  }    *//Using Lambda Expression----------------------------------*  Console.WriteLine("\n Output ( Using Lambda Expression ) : ");  data.ForEach(d => Console.WriteLine("\t- {0} -", d));  Console.ReadLine();  }  }  } |
| **Output :** |
|  |

| **Assignment 8** |
| --- |
| **Write a C# program to declare a List<int> , read 5 values from the user and find the sum using for, foreach and lambda expression loop types** |
| **Code :** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace listint3looptypes  {  internal class Program  {  static void Main(string[] args)  {  List<int> data = new List<int>();  int temp;  int sum1 = 0, sum2 = 0, sum3 = 0;  *//Declaring Values to List----------------------------------------*  for (int i = 0; i <= 4; i++)  {  Console.Write("Enter any value into the index {0}, into the List : ", i);  temp = Convert.ToInt32(Console.ReadLine());  data.Add(temp);  }  *//Using For Loop--------------------------------------------------*  Console.WriteLine("\n\nOutput ( Using For Loop ) : ");  Console.WriteLine("----------------------------------------------------------------------------");  for (int i = 0; i < data.Count; i++)  {  sum1 = sum1 + data[i];  }  Console.WriteLine("\n The sum is : {0}", sum1);  *//Using Foreach Loop----------------------------------------------*  Console.WriteLine("\n\nOutput ( Using Foreach Loop ) : ");  Console.WriteLine("----------------------------------------------------------------------------");  foreach (var d in data)  {  sum2 = sum2 + d;  }  Console.WriteLine("\n The sum is : {0}", sum2);  *//Using Lambda Expression-----------------------------------------*  Console.WriteLine("\n\nOutput ( Using Lambda Expression ) : ");  Console.WriteLine("----------------------------------------------------------------------------");  data.ForEach(d => sum3 = sum3 + d);  Console.WriteLine("\n The sum is : {0}", sum3);  Console.ReadLine();  }  }  } |
| **Output :** |
|  |

| **Assignment 9** |
| --- |
| **In a tabular format , write all data types in C# and write the respective alias names** |
| **Answer :** |
| | **Datatype** | **Alias Name** | **Class Name** | | --- | --- | --- | | byte | Byte | System.Byte | | ushort | Uint16 | System.Uint16 | | uint | Uint32 | System.Uint32 | | ulong | Uint64 | System.Uint64 | | sbyte | SByte | System.Sbyte | | short | Int16 | System.Int16 | | int | Int32 | System.Int32 | | long | Int64 | System.Int64 | | float | Single | System.Single | | double | Double | System.Double | | decimal | Decimal | System.Decimal | | bool | Boolean | System.Boolean | | char | Char | System.Char | | string | String | System.String | |
| **Output :** |
|  |

| **Assignment 10** |
| --- |
| **Write a C# program for implicit and explicit type casting** |
| **Code :** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace imp\_exp\_conversion  {  internal class Program  {  static void Main(string[] args)  {  *// implicit conversion*  *// Type Casting short to int*  short newShort = 8;  int newInt1 = newShort;  Console.WriteLine("Implicit Conversion of short to int is : {0}", newInt1);  *// Explicit Conversion*  *// Type*  *Casting double to int*  double newDouble = 11.23;  int newInt2 = (int)newDouble;  Console.WriteLine("Explicit Conversion of double to int is : {0}", newInt2);  Console.ReadLine();  }  }  } |
| **Output :** |
|  |