**1)Size of an array is static and it can not be altered once declared.**

**2)** **using System;**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**int day = 1;**

**string dayString;**

**switch (day)**

**{**

**case 1:**

**case 2:**

**dayString = "Monday";**

**break;**

**case 3:**

**dayString = "Tuesday";**

**break;**

**case 4:**

**dayString = "Wednesday";**

**break;**

**case 5:**

**dayString = "Thursday";**

**break;**

**case 6:**

**dayString = "Friday";**

**break;**

**case 7:**

**dayString = "Saturday";**

**break;**

**default:**

**dayString = "Invalid day";**

**break;**

**}**

**Console.WriteLine(dayString);**

**/\***

**The above code displays 'Monday' as output for day=1.**

**Change day=2, again the output is 'Monday'.**

**If no statement is written inside the satisfied switch case**

**then the output is considered from the very next switch case.**

**\*/**

**3)**

**Note : Array.Rank Property gets the rank (number of dimensions) of the Array.**

**For example, a one-dimensional array returns 1, a two-dimensional array returns 2, and so on.**

4) **To pass the parameters in any order other than the order specified in the method definition, 'Named parameters' can be used.**

**To set default values for method parameters, 'Optional parameters' can be used to while calling the method. Defining 'Optional parameters' allows us to omit passing values for some formal parameters.**

**Go through the below code and observe named and optional parameter used for its implementation. Execute and observe the output.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

public class Purchase

{

public void PrintPurchaseDetails(string purchaseId, string productName,

int quantityOrdered, string shippingAddress = "NA")

{

Console.WriteLine("Purchase Details are :");

Console.WriteLine("----------------------------------------------");

Console.WriteLine("Purchase Id = {0}", purchaseId);

Console.WriteLine("Product Name = {0}", productName);

Console.WriteLine("Quantity Ordered = {0}", quantityOrdered);

Console.WriteLine("Shipping Address = {0}", shippingAddress);

Console.WriteLine("----------------------------------------------");

}

}

class Program

{

static void Main(string[] args)

{

Purchase purchaseOne = new Purchase();

// Method invocation using positional parameters

// Values are not passed using their parameter names

Console.WriteLine("Normal method invocation");

purchaseOne.PrintPurchaseDetails("100001", "Coke", 20, "3rd cross,M.G road, Bangalore");

// Positional arguments should be given before named parameters

// Named parameters - Values are passed using their parameter names

Console.WriteLine("Method invocation with positional parameters followed by named parameters");

purchaseOne.PrintPurchaseDetails("100001", "Coke", quantityOrdered: 20, shippingAddress: "3rd cross,M.G road, Bangalore");

// Uncomment the below line and observe a compilation error -positional arguments cannot be after named parameters

//purchaseOne.PrintPurchaseDetails( quantityOrdered: 20, shippingAddress: "3rd cross,M.G road, Bangalore", "100001", "Coke");

// Uncomment the below line and observe positional arguments mixed with named parameters results in a compiler error

//purchaseOne.PrintPurchaseDetails( "100001",quantityOrdered: 20, shippingAddress: "3rd cross,M.G road, Bangalore", "Coke");

// Method invocation using only named parameters

Console.WriteLine("Method invocation using named parameters");

purchaseOne.PrintPurchaseDetails(productName: "Coke", purchaseId: "100001", shippingAddress: "3rd cross,M.G road, Bangalore", quantityOrdered: 20);

// Method invocation using optional parameters

// ShippingAddress is omitted in the method invocation

Console.WriteLine("Method invocation using optional parameters");

purchaseOne.PrintPurchaseDetails("100001", "iPhone5", 20);

}

}