GR 22 Regulations
II B.Tech II Semester
Visual Programming Using C# and .Net Lab
(GR22A2080)

GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

SYLLABUS

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY
VISUAL PROGRAMMING USING C# AND .NET LAB

Course Code: GR22A2080
L/T/P/C:0/0/4/2

II Year II Semester

Course Objectives:
1. To provide hands on experience on .Net framework.
2. To appreciate the asynchronous event handling feature in .Net.
3. To offer end-to-end program model for web application development.
4. To develop applications for the .NET Framework using C#
5. To learn C# debugging techniques

Course Outcomes:

1. Create Event Driven Applications.
2. Develop asynchronous applications
3. Deploy Web services
4. Build database applications using ADO.NET
5. Understand the Language Integrated Query (Linq) library

TASK1: Write a program to check weather a given number is palindrome using C#

TASK2: Write a program to implement the concept of Overloading in C#

TASK3: Write a program to store the employee details using class and methods in C#

TASK4: Write a program to implement single level and multilevel inheritance in C#

TASK5: Create a Window Form using HTML Controls

TASK6: Demonstrate basic string manipulation operations using String Builder and String classes in C#

TASK7: Demonstrate the concept of

a) Creating a Thread
b) Managing a Thread
c) Deleting a Thread

TASK8: Create a Sample program to Demonstrate insertion of data into database.

TASK9: Create a Program to Demonstrate Color Dialog in C#.

TASK10: Create a program to perform validation using validation controls.

TASK11: Create a Sample program to Demonstrate creation and usage of Dynamic Link Libraries in C#.

TASK12: Implement Student Management System with required details. Use ADO.NET for storing and manipulating the data. Develop the necessary forms for the better user interface.

TASK 1: Write a program to check weather a given number is palindrome or not in C#

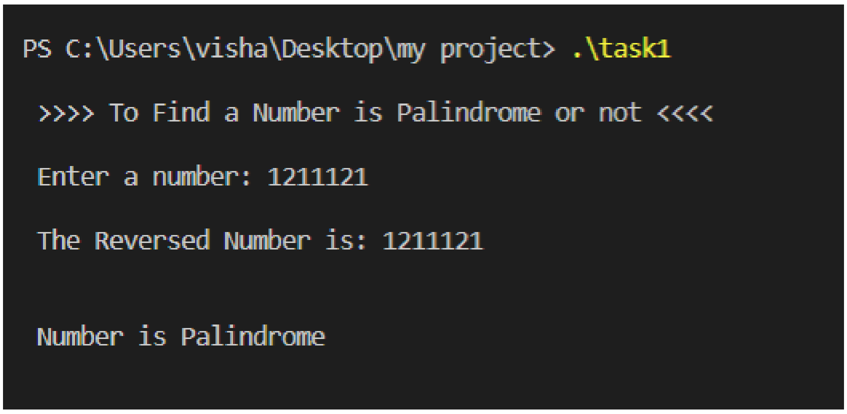
```
using System;
  public class P
   {
     public static void Main (string [] args)
      {
          int n,r,sum=0,temp;
          Console.Write("Enter the Number: ");
          n = int.Parse(Console.ReadLine());
          temp=n;
          while(n>0)
          {
           r=n%10;
           sum=(sum*10)+r;
           n=n/10;
          }
          if(temp==sum)
           Console.Write("Number is Palindrome.");
          else
           Console.Write("Number is not Palindrome");
    }
  }
```

STEP1: After creating new csc file with class name **P.cs** save above program

STEP2: Open new terminal and type command:csc P.cs

STEP3: then type: .\P

OUTPUT:

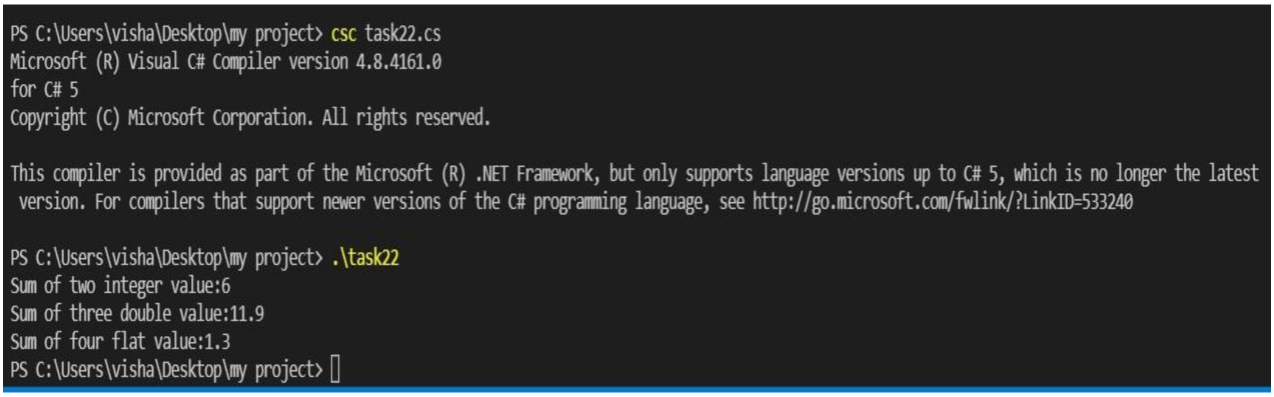


```
PS C:\Users\visha\Desktop\my project> .\task1

 >>>> To Find a Number is Palindrome or not <<<<

 Enter a number: 1211121

 The Reversed Number is: 1211121


 Number is Palindrome
```

TASK 2: Write a program to implement the concept of Overloading in C#

```
using System;

public class Methodoverloading
  {
    public int add(int a, int b)  //two int type Parameters method
    {
int sum=a+b;
            return sum;
    }
    public double add(double a, double b,double c)
    {
        double sum=a+b+c;
        return sum;

    }
    public float add(float a, float b,float c,float d)
    {
        float sum=a+b+c+d;
        return sum;

    }

public static void Main(string[] args)
{
Methodoverloading ob=new Methodoverloading();

int sum2=ob.add(2,4);
Console.WriteLine("Sum of two integer value:"+sum2);

double sum3=ob.add(5.2d, 3.5d,3.2d);
Console.WriteLine("Sum of three double value:"+sum3);

float sum4=ob.add(0.1f,0.3f,0.8f,0.1f);
Console.WriteLine("Sum of four flat value:"+sum4);
}
}
```

OUTPUT:



```
PS C:\Users\visha\Desktop\my project> csc task22.cs
Microsoft (R) Visual C# Compiler version 4.8.4161.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports language versions up to C# 5, which is no longer the latest
 version. For compilers that support newer versions of the C# programming language, see http://go.microsoft.com/fwlink/?LinkID=533240

PS C:\Users\visha\Desktop\my project> .\task22
Sum of two integer value:6
Sum of three double value:11.9
Sum of four flat value:1.3
PS C:\Users\visha\Desktop\my project> 
```

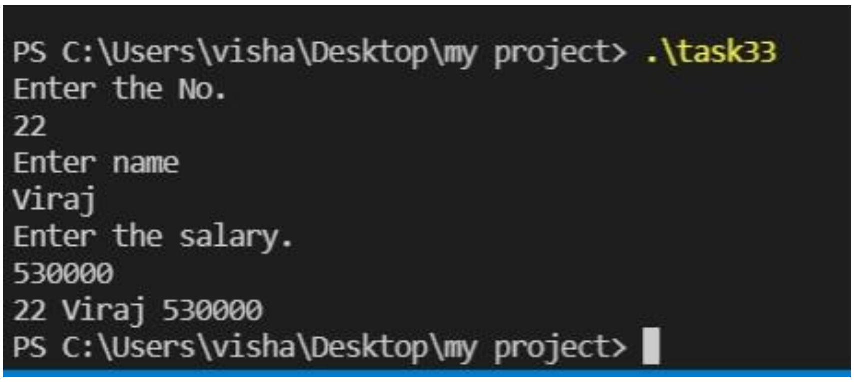
TASK 3: Write a program to store the employee details using class and methods in C#

```
using System;

 public class employee
{
    public int id;
    public string name;
    public float salary;
    public void insert(int i, string n, float s)
    {
        id=i;
        name=n;
        salary=s;
    }
    public void display()
    {
        Console.WriteLine(id+ " " +name+ " " +salary);

    }
}

class test
{
   public static void Main(string[] args)
    {
        employee e1 = new employee();

         int a;
         string b;
         float c;
         Console.WriteLine("Enter the No.");
        a=Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Enter name");
        b=Convert.ToString(Console.ReadLine());
        Console.WriteLine("Enter the salary.");
        c=Convert.ToInt32(Console.ReadLine());
        e1.insert(a,b,c);

         e1.display();


    }
}
```

STEP1: csc Employee.cs

STEP2: .\Employee.exe

OUTPUT:

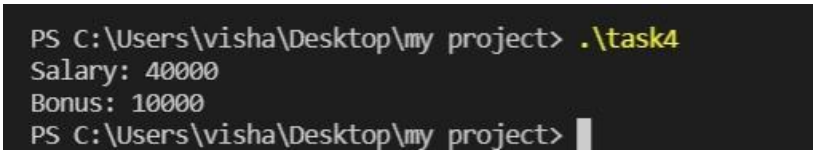


```
PS C:\Users\visha\Desktop\my project> .\task33
Enter the No.
22
Enter name
Viraj
Enter the salary.
530000
22 Viraj 530000
PS C:\Users\visha\Desktop\my project>
```

TASK 4: Write a program to implement single level and multilevel inheritance in C#

```
using System;
  public class Employee
   {
      public float salary = 40000;
  }
  public class Programmer: Employee
  {
      public float bonus = 10000;
  }
  class TestInheritance{
      public static void Main(string[] args)
       {
          Programmer p1 = new Programmer();


          Console.WriteLine("Salary: " + p1.salary);
          Console.WriteLine("Bonus: " + p1.bonus);


       }
   }
```

OUTPUT:



```
PS C:\Users\visha\Desktop\my project> .\task4
Salary: 40000
Bonus: 10000
PS C:\Users\visha\Desktop\my project>
```

TASK 5: Create a Window Form using HTML Controls

Step1: Check the dotnet version in the command Prompt by using below command

>dotnet –version

Step2: A new folder is to be created,

>dotnet new webapp -o MyWebApp --no-https -f net6.0

The dotnet new command creates a new application.

- The webApp parameter selects what template to use when creating your app.
- The -o parameter creates a directory named MyWebApp where your app is stored.
- The --no-https flag specifies not to enable HTTPS.
- The -f parameter indicates you're creating a .NET 6 application.

Several files were created in the MyWebApp directory to give you a simple web application that is ready to run.

- Program.cs contains the app startup code and middleware configuration.
- The Pages directory contains some example web pages for the application.
- MyWebApp.csproj defines some project settings, such as, .NET SDK version to target.
- The launchSettings.json file inside the Properties directory defines different profile settings for the local development environment. A port number ranging between 5000-5300 is automatically assigned at project creation and saved on this file.

A new folder with name MyWebApp is created in C:\Users\visha\MyWebApp

Step3: MyWebApp →Pages→Index(cshtml source file) in this notepad

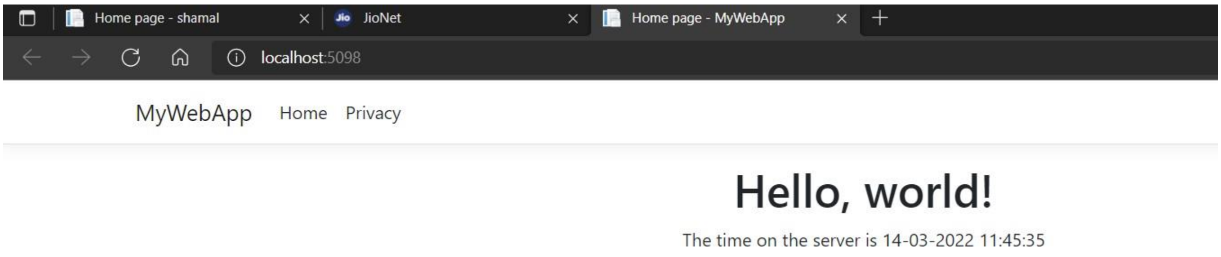
Pages/Index.cshtml

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1>Hello, world!</h1>
    <p>The time on the server is @DateTime.Now</p>
</div>
```

Step4: again, goto cmd prompt and type commands:

```
cd MyWebApp
dotnet watch
```

Step5: A Output is displayed on the Window page(browser).



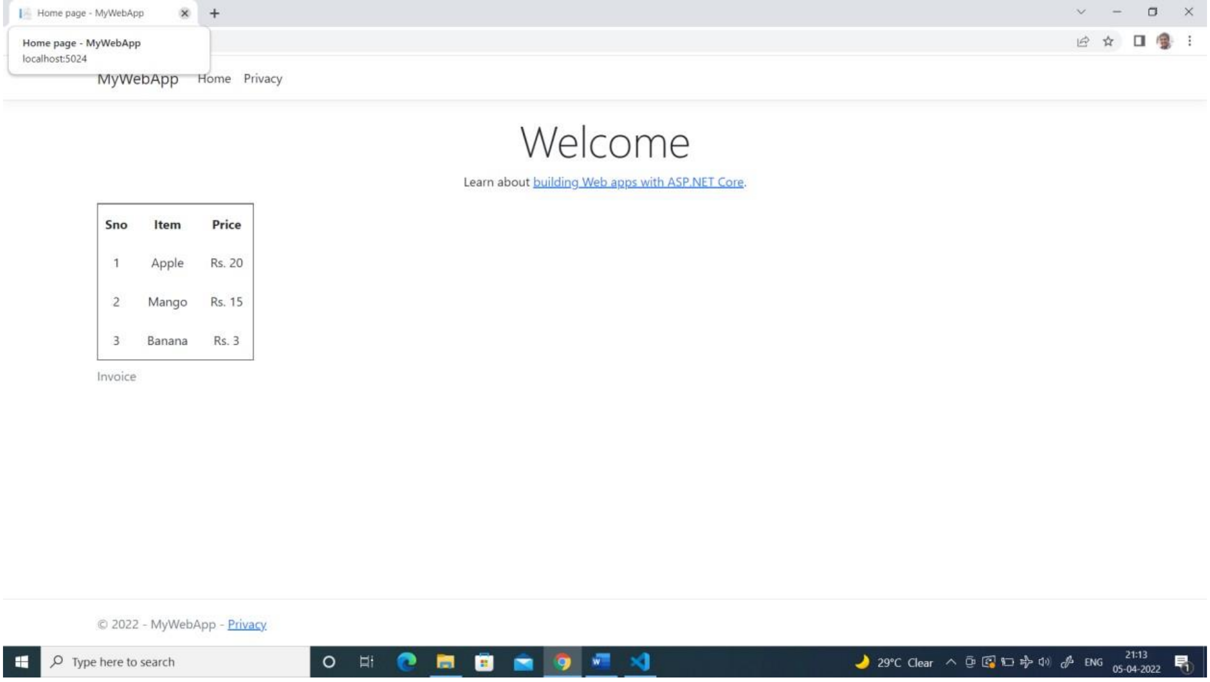
Hello, world!

The time on the server is 14-03-2022 11:45:35

EXAMPLE PROGRAM

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a></p>
    <head>
<title>Table</title>
</head>
<body>
<table border="1" width="200" height="200">
<caption>Invoice</caption>
<tr>
    <th>Sno</th>
    <th>Item</th>
    <th>Price</th>
</tr>
<tr>
    <td>1</td>
    <td>Apple</td>
    <td>Rs. 20</td>
</tr>
<tr>
    <td>2</td>
    <td>Mango</td>
    <td>Rs. 15</td>
</tr>
<tr>
    <td>3</td>
    <td>Banana</td>
    <td>Rs. 3</td>
</tr>
</table>
</body>
</div>
```

OUTPUT:



Welcome

Learn about building Web apps with ASP.NET Core.

Sno	Item	Price
1	Apple	Rs. 20
2	Mango	Rs. 15
3	Banana	Rs. 3

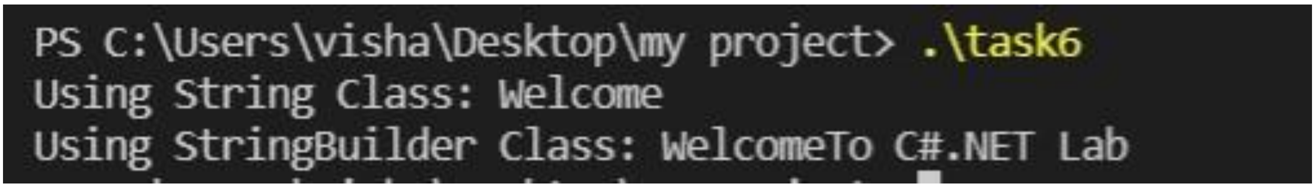
Invoice

© 2022 - MyWebApp - Privacy

TASK6: Demonstrate basic string manipulation operations using String Builder and String classes in C#

```
// C# program to demonstrate the
// difference between String,
// StringBuilder

using System;
using System.Text;
using System.Collections;

class stringbuilder {
    // Concatenates to String
    public static void concat1(String s1)
    {

        // taking a string which is to be Concatenated
        String st = "To C#.NET Lab";

        // using String.Concat method
        // you can also replace it with
        // s1 = s1 + "To C#.NET Lab";
        s1 = String.Concat(s1, st);
    }

    // Concatenates to StringBuilder
    public static void concat2(StringBuilder s2)
    {

        // using Append method of StringBuilder class
        s2.Append("To C#.NET Lab");
    }

    // Main Method
    public static void Main(String[] args)
    {

        String s1 = "Welcome";
        concat1(s1); // s1 is not changed
        Console.WriteLine("Using String Class: " + s1);

        StringBuilder s2 = new StringBuilder("Welcome");
        concat2(s2); // s2 is changed
        Console.WriteLine("Using StringBuilder Class: " + s2);
    }
}
```

OUTPUT:



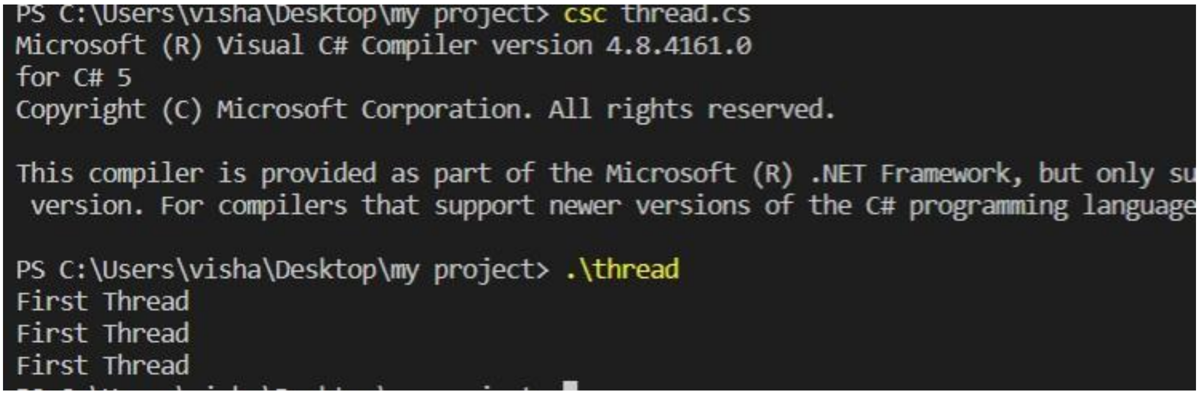
```
PS C:\Users\visha\Desktop\my project> .\task6
Using String Class: Welcome
Using StringBuilder Class: WelcomeTo C#.NET Lab
```

TASK 7: Demonstrate the concept of

a) **Creating a Thread**
b) **Managing a Thread**
c) **Deleting a Thread**

```
using System;
 using System.Threading;

 public class ExThread {

     // Non-static method
     public void mythread1()
     {
         for (int z = 0; z < 3; z++) {
             Console.WriteLine("First Thread");
         }
     }
 }

 // Driver Class
 public class thread {

     // Main Method
     public static void Main()
     {
         // Creating object of ExThread class
         ExThread obj = new ExThread();

         // Creating thread
         // Using thread class
         Thread thr = new Thread(new ThreadStart(obj.mythread1));
         thr.Start();

 // To Delete/Abort a Thread
 // Console.WriteLine(“Thread is abort”);
 //thr.Abort();

     }
 }
```

OUTPUT:



```
PS C:\Users\visha\Desktop\my project> csc thread.cs
Microsoft (R) Visual C# Compiler version 4.8.4161.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only su
 version. For compilers that support newer versions of the C# programming language

PS C:\Users\visha\Desktop\my project> .\thread
First Thread
First Thread
First Thread
```

Task 8: Create a Sample program to Demonstrate insertion of data into database.

Step1: Instal mysql using Oracle login credentials.

Step2: Open advance settings to set the environment variable (set this particular **path** : C:\Program Files\MySQL\MySQL Server 8.0\bin)

Step3: Open the command prompt and run the commands as shown below

Cmd1: mysql -–version

Cmd2: mysql -u root -p

Example (pwd: ********)

Provide the password that is set to the root user.

Query: create database myshop;

Query: create table myshop.table1(slnum int,Name varchar(20));

After table is created

Create a folder>> Open it using vs code>>Then open terminal and type

Cmd1:cd folder name

Cmd2: dotnet new console

program.cs

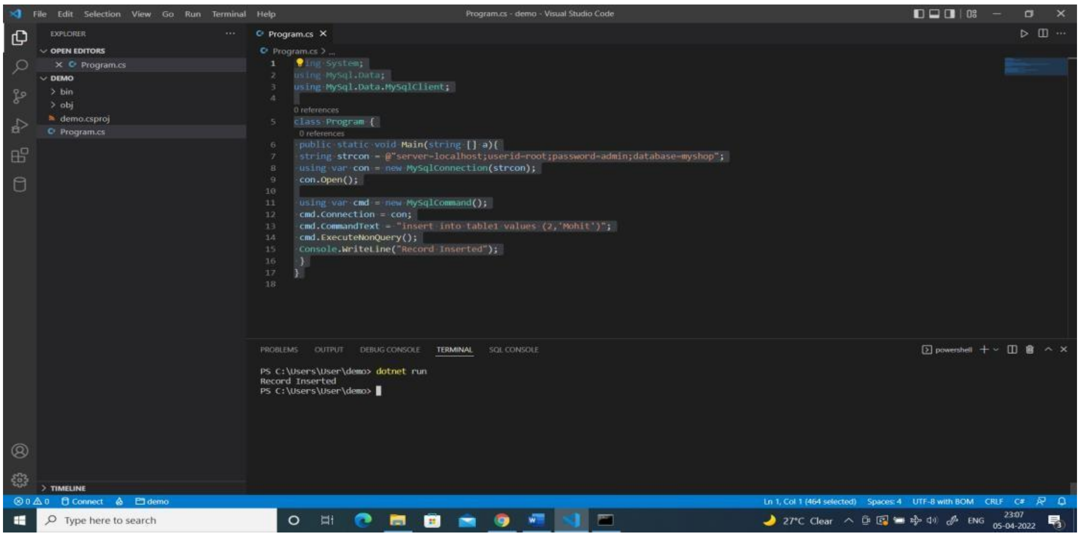
```
using System;

using MySql.Data;

using MySql.Data.MySqlClient;

class Program {

 public static void Main(string [] a){

 string strcon = @"server=localhost;userid=root;password=admin;database=myshop";

 using var con = new MySqlConnection(strcon);

 con.Open();

 using var cmd = new MySqlCommand();

 cmd.Connection = con;
 cmd.CommandText = "insert into table1 values (2,'Mohit')";

 cmd.ExecuteNonQuery();

 Console.WriteLine("Record Inserted");

 }

}
```

Before running above program also install mysql packages by using command

```
>dotnet add package Mysql.Data
```

finally, run the code

```
>dotnet run
```



```
ing System;
using MySql.Data;
using MySql.Data.MySqlClient;

class Program {
 public static void Main(string [] a){
 string strcon = @"server=localhost;userid=root;password=admin;database=myshop";
 using var con = new MySqlConnection(strcon);
 con.Open();

 using var cmd = new MySqlCommand();
 cmd.Connection = con;
 cmd.CommandText = "insert into table1 values (2,'Mohit')";
 cmd.ExecuteNonQuery();
 Console.WriteLine("Record Inserted");
 }
}
```

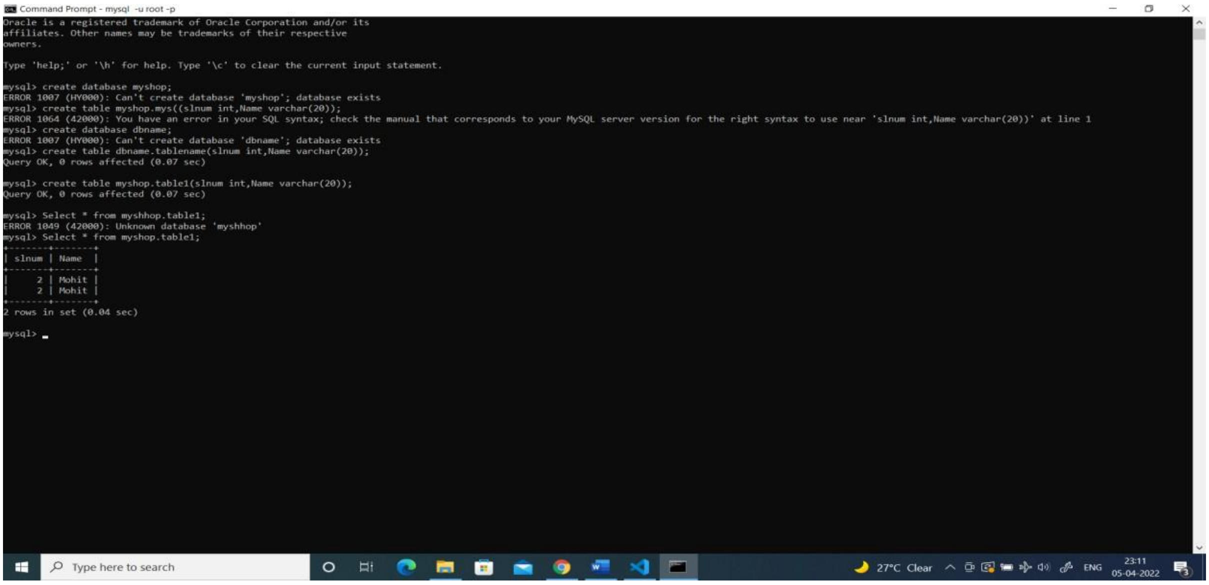
```
PS C:\Users\User\demo> dotnet run
Record Inserted
PS C:\Users\User\demo>
```

Cross check whether the record is inserted or not using

Select * from db_name.table_name;

In our case

Select * from myshop.table1;



```
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database myshop;
ERROR 1007 (HY000): Can't create database 'myshop'; database exists
mysql> create table myshop.mys((slnum int,Name varchar(20));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'slnum int,Name varchar(20))' at line 1
mysql> create database dbname;
ERROR 1007 (HY000): Can't create database 'dbname'; database exists
mysql> create table dbname.tablename(slnum int,Name varchar(20));
Query OK, 0 rows affected (0.07 sec)

mysql> create table myshop.table1(slnum int,Name varchar(20));
Query OK, 0 rows affected (0.07 sec)

mysql> Select * from myshhop.table1;
ERROR 1049 (42000): Unknown database 'myshhop'
mysql> Select * from myshop.table1;
+-------+-------+
| slnum | Name  |
+-------+-------+
|     2 | Mohit |
|     2 | Mohit |
+-------+-------+
2 rows in set (0.04 sec)

mysql> _
```

Task 9: Create a Program to Demonstrate Color Dialog in C#.

Step1: open visual studio code new terminal

Step2: run cmd

```
>dotnet new winforms -o first
```

Step3: three file editors be created at place of **first**(editor)here **first** is a folder been created,

With file names

form1.cs

form1designer.cs

program.cs

Step4:we can remove form1.cs and form1designer.cs by right clicking on the file and click delete

Step5:write the program below in program.cs

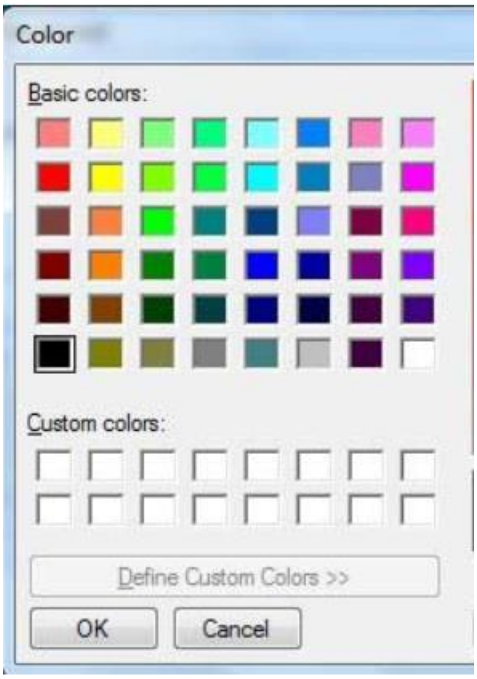
Final step:Save the above program and run command in terminal:

```
dotnet run --project first
```

After above command colour dialog box be visible

```
using System.Windows.Forms;
using System.Drawing;
namespace First
{
public class MyForm : Form
 {
public MyForm()
{
 InitComponents();
 }
private void InitComponents()
{
ColorDialog dlg = new ColorDialog();
dlg.ShowDialog();


}


 [STAThread]
 static void Main() {
Application.SetHighDpiMode(HighDpiMode.SystemAware);
Application.EnableVisualStyles();
Application.Run(new MyForm());
}
 }
}
```

OUTPUT:


Color
Basic colors:
Custom colors:
Define Custom Colors >>
OK
Cancel

TASK 10: Create a program to perform validation using validation controls.

ASP.NET validation controls validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get stored.

ASP.NET provides the following validation controls:

- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator
- ValidationSummary

BaseValidator Class

The validation control classes are inherited from the BaseValidator class hence they inherit its properties and methods. Therefore, it would help to take a look at the properties and the methods of this base class, which are common for all the validation controls:

Members	Description
ControlToValidate	Indicates the input control to validate.
Display	Indicates how the error message is shown.
EnableClientScript	Indicates whether client side validation will take.
Enabled	Enables or disables the validator.
ErrorMessage	Indicates error string.
Text	Error text to be shown if validation fails.
IsValid	Indicates whether the value of the control is valid.
SetFocusOnError	It indicates whether in case of an invalid control, the focus should switch to the related input control.
ValidationGroup	The logical group of multiple validators, where this control belongs.

Validate()	This method revalidates the control and updates the IsValid property.

RequiredFieldValidator Control

The RequiredFieldValidator control ensures that the required field is not empty. It is generally tied to a text box to force input into the text box.

The syntax of the control is as given:

```
<asp:RequiredFieldValidator ID="rfvcandidate"
  runat="server" ControlToValidate ="ddlcandidate"
  ErrorMessage="Please choose a candidate"
  InitialValue="Please choose a candidate">
  </asp:RequiredFieldValidator>
```

RangeValidator Control

The RangeValidator control verifies that the input value falls within a predetermined range.

It has three specific properties:

Properties	Description
Type	It defines the type of the data. The available values are: Currency, Date, Double, Integer, and String.
MinimumValue	It specifies the minimum value of the range.
MaximumValue	It specifies the maximum value of the range.

The syntax of the control is as given:

```
<asp:RangeValidator ID="rvclass" runat="server" ControlToValidate="txtclass"
  ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
  MinimumValue="6" Type="Integer">
  </asp:RangeValidator>
```

CompareValidator Control

The CompareValidator control compares a value in one control with a fixed value or a value in another control.

It has the following specific properties:

Properties	Description
Type	It specifies the data type.
ControlToCompare	It specifies the value of the input control to compare with.
ValueToCompare	It specifies the constant value to compare with.
Operator	It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck.

The basic syntax of the control is as follows:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
  ErrorMessage="CompareValidator">
  </asp:CompareValidator>
```

RegularExpressionValidator

The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression. The regular expression is set in the ValidationExpression property.

The following table summarizes the commonly used syntax constructs for regular expressions:

Character Escapes	Description
\b	Matches a backspace.
\t	Matches a tab.
\r	Matches a carriage return.
\v	Matches a vertical tab.
\f	Matches a form feed.
\n	Matches a new line.
\	Escape character.

Apart from single character match, a class of characters could be specified that can be matched, called the metacharacters.

Metacharacters	Description
.	Matches any character except \n.
[abcd]	Matches any character in the set.
[^abcd]	Excludes any character in the set.
[2-7a-mA-M]	Matches any character specified in the range.
\w	Matches any alphanumeric character and underscore.
\W	Matches any non-word character.
\s	Matches whitespace characters like, space, tab, new line etc.
\S	Matches any non-whitespace character.
\d	Matches any decimal character.
\D	Matches any non-decimal character.

Quantifiers could be added to specify number of times a character could appear.

Quantifier	Description
*	Zero or more matches.
+	One or more matches.
?	Zero or one matches.
{N}	N matches.
{N,}	N or more matches.
{N,M}	Between N and M matches.

The syntax of the control is as given:

```
<asp:RegularExpressionValidator ID="string" runat="server" ErrorMessage="string"
  ValidationExpression="string" ValidationGroup="string">
  </asp:RegularExpressionValidator>
```

CustomValidator

The CustomValidator control allows writing application specific custom validation routines for both the client side and the server side validation.

The client side validation is accomplished through the ClientValidationFunction property. The client side validation routine should be written in a scripting language, such as JavaScript or VBScript, which the browser can understand.

The server side validation routine must be called from the control's ServerValidate event handler. The server side validation routine should be written in any .Net language, like C# or VB.Net.

The basic syntax for the control is as given:

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
  ClientValidationFunction=.cvf_func. ErrorMessage="CustomValidator">
  </asp:CustomValidator>
```

ValidationSummary

The ValidationSummary control does not perform any validation but shows a summary of all errors in the page. The summary displays the values of the ErrorMessage property of all validation controls that failed validation.

The following two mutually inclusive properties list out the error message:

ShowSummary : shows the error messages in specified format.

ShowMessageBox : shows the error messages in a separate window.

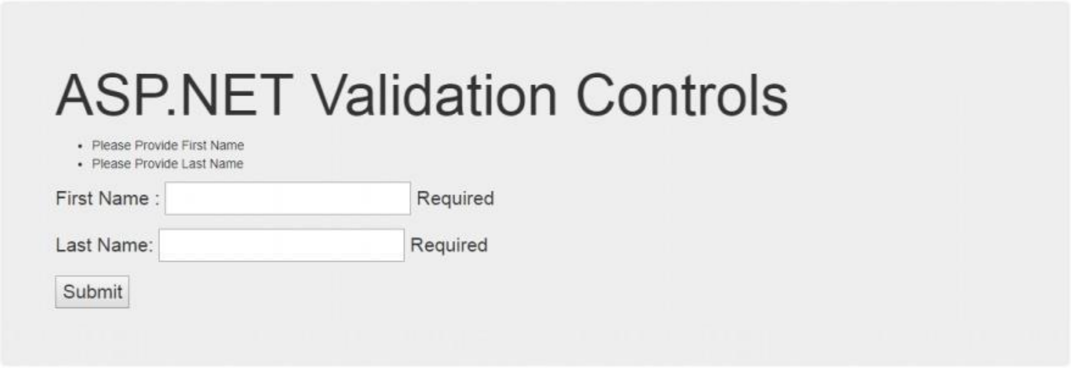
The syntax for the control is as given:

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
  DisplayMode = "BulletList" ShowSummary = "true" HeaderText="Errors:" />
```

```
<h1>ASP.NET Validation Controls</h1>
 <asp:ValidationSummary ID="ValidationSummary1" runat="server" />
 <p>
```

```
  First Name :
 <asp:TextBox ID="txtFirstName" runat="server" CausesValidation="True">
 </asp:TextBox>
 <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server"         ErrorMessage="Please Provide First
Name"  ControlToValidate="txtFirstName"> Required
</asp:RequiredFieldValidator>
</p>
<p>
Last Name:
 <asp:TextBox ID="txtLastName" runat="server">
 </asp:TextBox>
 <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
ErrorMessage="Please Provide Last Name" ControlToValidate="txtLastName"> Required
 </asp:RequiredFieldValidator>
</p>
<p>
<asp:Button ID="Button1" runat="server" Text="Submit" />
</p>
```

OUTPUT:



ASP.NET Validation Controls

- Please Provide First Name
- Please Provide Last Name

TASK 11: Create a Sample program to Demonstrate creation and usage of Dynamic Link Libraries in C#.

Filename1: Mymathlib.cs

```
using System;
namespace mymathlib
{
   public class method1
    {
      public static long Add(long x, long y)
      {
         return(x+y);

      }
      public static long Mul(long x, long y)
      {
         return(x*y);
      }
   }
}
```

Filename2:MyMathApp.cs

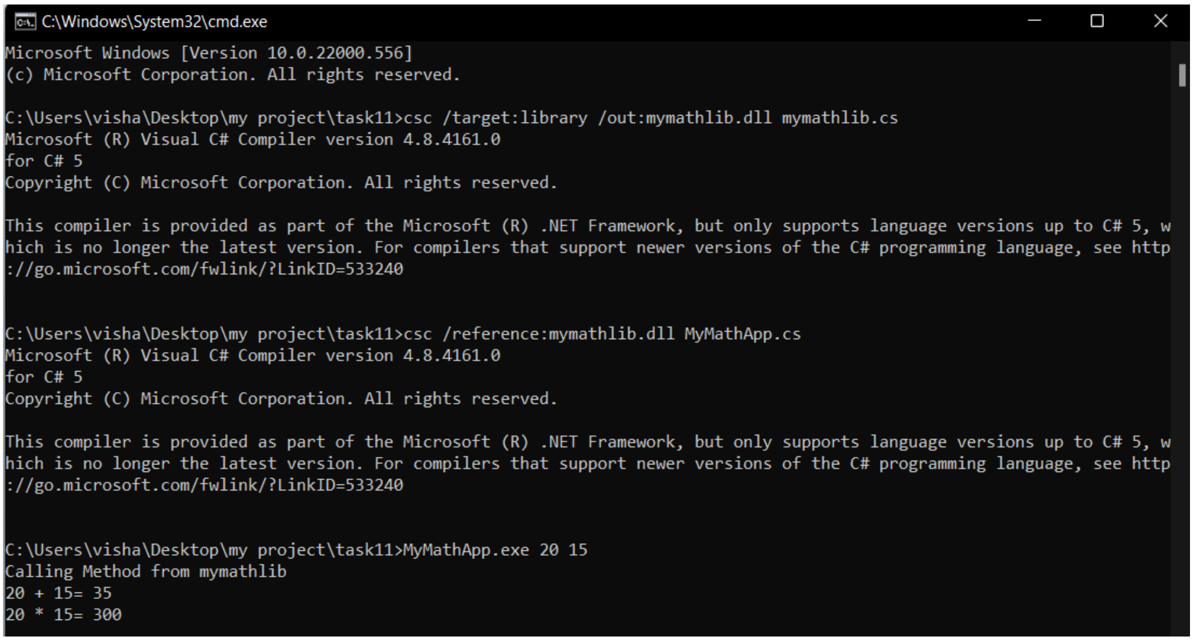
```
using System;
using mymathlib;

class  MyMathApp
{
   public static void Main(string[] args)
   {
      Console.WriteLine("Calling Method from mymathlib");

      if(args.Length !=2)
      {
         Console.WriteLine("Usage :MyMathApp <num1> <num2>");
         return;
      }

      long num1=long.Parse(args[0]);
      long num2=long.Parse(args[1]);

      long add=method1.Add(num1, num2);
      long mul=method1.Mul(num1, num2);

      Console.WriteLine( "{0} + {1}= {2}", num1,num2,add);
```

```
        Console.WriteLine( "{0} * {1}= {2}", num1,num2,mul);

    }
}
```

Command Prompt:

```
>csc /target:library /out:mymathlib.dll mymathlib.cs

> csc /reference:mymathlib.dll MyMathApp.cs

> MyMathApp.exe 20 15

> Calling Method from mymathlib

20 + 15= 35

20 * 15= 300
```

OUTPUT:



```
Microsoft Windows [Version 10.0.22000.556]
(c) Microsoft Corporation. All rights reserved.

C:\Users\visha\Desktop\my project\task11>csc /target:library /out:mymathlib.dll mymathlib.cs
Microsoft (R) Visual C# Compiler version 4.8.4161.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports language versions up to C# 5, w
hich is no longer the latest version. For compilers that support newer versions of the C# programming language, see http
://go.microsoft.com/fwlink/?LinkID=533240


C:\Users\visha\Desktop\my project\task11>csc /reference:mymathlib.dll MyMathApp.cs
Microsoft (R) Visual C# Compiler version 4.8.4161.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports language versions up to C# 5, w
hich is no longer the latest version. For compilers that support newer versions of the C# programming language, see http
://go.microsoft.com/fwlink/?LinkID=533240


C:\Users\visha\Desktop\my project\task11>MyMathApp.exe 20 15
Calling Method from mymathlib
20 + 15= 35
20 * 15= 300
```

TASK 12: Implement Student Management System with required details. Use ADO.NET for storing and manipulating the data. Develop the necessary forms for the better user interface.

Step 1: Create a database

1. Open **Visual Studio**.
2. Go to **Server Explorer**. If Server Explorer is not opened go to **View** Server **Explorer**.
3. Right click on **Data Connections** and select **Create New SQL Server Database**.
4. Fill details like this.

a.ServerName: **.\SQLEXPRESS**
b.Select **WindowsAuthentication**
c.DatabaseName: **TestDB**
d. Click **OK**.

Step 2:Create Table

1. Now, expand your database Right click on Tables Add New Table.

2. Create a new table like this.

```
CREATE TABLE [dbo].[Student] (

    [Id] INT NOT NULL,

    [Name] NVARCHAR (50) NULL,

    [Subject] NVARCHAR (50) NULL,

    PRIMARY KEY CLUSTERED ([Id] ASC)

);
```

3. After finishing table click on **Update** button that is the upper right corner of design pane.
4. Right click on Student Table and select **Show Table Data**. Now, fill some demo data in it like this.

Step 3: Get Connection String

1. Now, your database and tables are ready. You can find your connection string in propertiesofthedatabase.
a.RightClickon **Database Properties**

Now, you have a database with a table and a connection string. The next step is to connect to the database and retrieve the records using ADO.Net Commands.

INSERTING RECORD

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using Vertica.Data.VerticaClient;
namespace ConsoleApplication
{
    class Program
    {
        Public static void Main (string[] args)
        {
            VerticaConnectionStringBuilder builder = new VerticaConnectionStringBuilder();
                VerticaConnection _conn = new VerticaConnection(builder.ToString());
            _conn.Open();
            VerticaCommand command = _conn.CreateCommand();
                command.CommandText =
         "INSERT into test values(2, 'username', ‘subject’)";
            Int32 rowsAdded = command.ExecuteNonQuery();
            Console.WriteLine( rowsAdded + " rows added!");
           _conn.Close();
}
}
}
```

RETRIEVE RECORD

Start New Console Application.

1. Go to **File** > **New** > **Project**. Create a new console application, **FirstProgram** and click **OK**.

2. Paste the following code in the program window and press **ctrl + F5**

```
using System;
using System.Data.SqlClient;


namespace FirstProgram
{
    class Program
    {
        Public static void Main(string[] args)
        {
string ConString = @"Data Source=.\SQLEXPRESS;Initial Catalog=TestDB;Integrated Security=True";
            SqlConnection con = new SqlConnection(ConString);
            string querystring = "Select * from Student";
            con.Open();
            SqlCommand cmd = new SqlCommand(querystring, con);
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
Console.WriteLine(reader[0].ToString() + " " + reader[1].ToString() + " " + reader[2].ToString());
            }
        }
    }
}
```