Getting started Exercises

1.1 Installing Visual studio and setting up the .NET Environment

1. Download visual studio.

- Visit the official website of visual studio:

https:// visual studio.microsoft.com/

- Click on the "download" button for community edition.

2. Run the installer

- Once the download is complete, run the installer

- Follow the on screen instructions to continue

3. Select the workloads

- In the Installer, you will see a list of workloads. Seleck ".NET desktop devolopment".

- Optionally, you can select other workloads if needed for future use, such as a "ASP.NET and web devolopment.

4. Install

- Click on the "instal" button to begin the installation process

- Wait for the installation to complete. This may take time depending on your internet speed and system performance.

5. Launch Visual Studio

- Once the installation is done, Launch visual studio from start menu (or) desktop shortcut.

Setting up the .NET Environment

1. Create a new project

-Click on "Create a new project" from visual studio start page

-select the templates "Console App (.NET Core)"

-Click "Next"

2. Configure your project

-Enter a name for your project

-Choose a location to save project

-Click "Create"

3 Verify the setup:

-Once the project is created, you will see "program.cs" file open in code editor

-Replace existing code with following

```
using system;
namespace HelloWorld
{
    class Program
    {
        static void main (String args[])
        {
            Console.WriteLine ("HelloWorld!"),
        }

    }
}
```

Press ctrl+F5 to run application

Verify the output window displays "HelloWorld!"

is Distance based on speed and time

d = v×t

Code:

```
using System;
namespace sample
{
  class Program
   {
     static void main ( String [] args)
     {
       // Prompt for speed input
       Console Writeline ("Enter speed of object (m/s): ");
       double speed = Convert ToDouble ( Console ReadLine());
       // Validate speed input
       if ( speed<=0)
        {
        Console.WriteLine ("invalid iput: Speed must be greater
         than zero");
        return;
        }
        // Prompt for time input
        Console .WriteLine ( "Enter the time duration(s): ");
        double time = Convert.ToDouble (Console ReadLine());
        // validate time input
        if( time <=0)
        {
          Console Writeline ("Invalid input : Time must be
          greater than zero");
          return;
        }
       // Calculate distance
       double distance = speed * time ;
```

```
            //Display the result
            Console.Writeline ("Distance travelled :. "+ distan
            " metric");
            Console.Readkey();
            }
      }
}
```

Actual output

```
Enter the speed of the object (m/s):
10.2
Enter the time duration (s).
8.4
Distance travelled : 85.68 meters
```

1.3 Cube root

Calculate cube root of given number

Code:

```
using System;
namespace Program
{
  static void main (String [] args)
  {
    //prompt for input
    Console Writeline ("Enter the number");
    string input = Console.Readline();
    //validate input and calculate cube root
    if ( double.Try Parse (input, out double number))
    {
        double cubeRoot = CalculateCubeRoot(number);
        Console.Write Line(" Cube root of (number) is (cubeRoot)"
        Console.Readkey();
    }
else
{
    Console.Writeline (" Invalid input : Please enter a valid
       number.");
    Console.Readkey();
  }
}
// Method to Calculate the cube root
static double Calculate CubeRoot (double number)
{
      return Math.Pow (number, 1.0/3.0);
 }
}
```

Actual Output

```
Enter the number : se4
invalid input : Please enter a valid number.
Enter the number : 64
Cube root of 64 is 4
```

1.4 Random Number Generator

Generate and display a sequence of random numbers

Code :

```
Using System;
namespace sample
{
  class Program
  {
      static void maul String ()args)
      {
      //create a new instanceof random class
      Random random = new Random();
```

```
//Get user inputs
console Write ("Write a number of random numbers to
 generate: ");
if (!int TryParse(Console ReadLine(), out int count)|| count
 <=0)
{
 Console WriteLine ("Invalid input: Please enter a positive integer"
 return;
}

 Console Write( "Enter the maximum value of the range: ");
 if (! int .TryParse( Console ReadLine(), out int minValue))
 {
   console.WriteLine(" Invalid input: Please enter a valid
   number.")
                                         ;
   return,
 }

 Console.Write ("Enter the maximum Value of the range:");
 if (!int. Try Parse ( Console. ReadLine(), out int maxValue))
   maxValue <= minValue
 {

   Console.WriteLine(" Invalid input: Maximum value must be
         greater than minimum value.");
   return;
 }

 // Display the generated random number
 Console. WriteLine ($"Generated {count} unique random
 numbers between {minValue} and {maxValue}:");
 int randcount=1;
 while (randCount <= count)
 {
   int randomNumber = random .Next (min Value ,maxValue+
   console .Write( random Number +" ");                  1);
   randCount++;
 }
```

```
        Console.Readkey();
      }
    }
  }
```

Actual Output

```
Enter the number of random numbers to generate:10
Enter the minimum value of the range:100
Enter the maximum value of the range:500
Generated 10 unique random numbers between 100 and
500:
281  278 168  393  121  102 358 282  171  430
```

1.5 Nullable Data Types

Demonstrate the use of Nullable datatypes. Nullable data types allao variables to have an additional value, null, which represents undefined or unknow value

```
using NullableDemo
{
   static void main (String []args)
   {
     int?  intVal1 = null;
     int?  intVal2 = 786;
     float? floatval1 = 3.14F;
     float? floatval2 = newfloat?();
     bool?  boolval = new bool?();
     Console Writeline (" Nullable : {0}, {1}", intVal1, intVal2);
     Console Writeline (" Nullable Floats. {0}, {1}", floatVal1,
        floatVal2);
     console writeline ("Nullable boolean : {0}", boolval);
     Console.Readkey();
   }
}
```

Actual Output

Nullable Integers: ,786

Nullable Floats : 3.14,

Nullable . boolean :

1.6 Permutations (nPr)

```
using system;

class Npr
{
    static   int   Calculate Factorial (int n)
    {
        int fact=1;
        for (int i=2;  i<=n ;  i++)
        {
            fact = fact*i;
        }
        return fact;
    }
    static   int  Calculate Npr (int n,int r)
    {
        int npr=0;
        int fact1=0;
        int fact2=0;
        fact1 = Calculate Fatorial (n);
        fact2 = Calculate Factorial (n-r);
        npr= fact1/fact2;
        return npr;
    }
    static  void  main (String [] args)
    {
        int npr=0;
        int n =0;
        int r =0;
Console Writ ("Enter the value of 'n': ");
n= int . Parse ( Console . Readline());
Console . Writ ("Enter the value of 'r' : ");
r= int . Parse(Console. Readline());
npr= Calculate Npr(n,r);
Console Write Line ( "Npr:" +npr);
Console . Read Key();
}
}
```

Actual output

Test Case 1:

```
Enter the value of 'n': 5
Enter the value of 'r':3
Npr:60
```

Test case 2:

```
Enter the value of 'n' : 10
Enter the value of 'r': 5
Npr : 30240
```

1.7 Binary sum

code:

```
class BinarySum
{
  static void Main()
   {
     Console .Write ("Enter 1st binary number:");
     string binary1= console . Readline();
     Console . Write (" Enter 2nd binary number:");
     string binary2 = Console . Read Line();
     if ( ! Is Binary (binary1) || ! Is Binary(binary2))
     {
       Console .WriteLine ("Error One or both inputs are not valid
                                binary number:");
```

```
        console Readkey(),
         return,
    }
    string  result  = Add Binary (binary1, binary2);
    Console. Write line ($"Sum  is : {result}"),
    Console. Readkey();
  }
 static bool IsBinary (String  binary)
 {
   foreach ( char  c  in  binary
    {
     if (c!='0' && c!='1')
      return false;
    }
     return true;
 }
 static string   Add Binary (string a, string b)
 {
   int maxlength= Math. max (a. length, b.length),
   a. a.Padleft( maxlength, '0'),
   b. b Padleft (maxlength, '0'),
   string result = "";
   int carry=0,
   for (int  i = maxlength -1 ;  i>=0 ; i--)
   {
      int  bitA = a[i]-'0';
      int  bitB = b[i] - '0';
      int  sum  = bitA + bitB + carry;
      result  =  (sum%2) + result;
      carry = sum/2,
}
if (carry>0)
result = carry + result;
 return result;
}
}
```

Actual Output

Test Case 1:

```
Enter 1st binary number: 101
Enter 2nd binary number: 1001
Sum is: 1110
```

1.8 Explore Bitwise Operators

Code.

```
using System;
class BitwiseOpertorsDemo
{
  static void Main()
  {
       Console.Write("Enter the first integer:");
       int num1 = Convert.ToInt32(Console.ReadLine());
       Console.Write("Enter the second integer:");
       int num2 = Convert.ToInt32(Console.ReadLine());
       int andResult = num1 & num2;
       int orResult = num1 | num2;
       int xorResult = num1 ^ num2;
       int notResult = ~num1;
       int leftshiftResult = num1<<2;
       int rightshiftResult = num1>>2;
       Console.WriteLine($"Bitwise AND{num1} & {num2}= {andRe
       Console.WriteLine($"Bitwise OR{num1}|{num2}= {orResult
        Console WriteLine ($" Bitwise XOR ( {num1} ^ {num2}) = {xorResult}");
        Console WriteLine ( $" Bitwise NOT (~{num1}) = {notResult} ");
        Console WriteLine ( $" Leftshift ( {num1}<<2) = {leftShiftResult}");
        Console WriteLine
                          ( $" Rightshift ({num1}>>2) = {rightShiftResult}");

        Console. Read Key();

    }
}
```

Actual Output

```
Enter the first integer: 12
Enter the second integer 10

Bitwise AND( 12 & 10) = 8
Bitwise OR (12 | 10) = 14
Bitwise XOR (12 ^ 10) = 6
Bitwise NOT (~12) = -13
Leftshift (12 <<2) = 48
Rightshift( 12 >>2) = 3
```

1.7 No Math

Code:

```
Using System;
class SquareRoot And Absolute Value
{
    ~~static void main (string []args~~
    static double Find AbsoluteValue (double number)
    {
      if (number < 0)
      number = numbe * -1;
      return number;
}
static double Calculate Square Root(doublenumber)
{
      if (number<0)
      Console.WriteLine ("Cannot Calculate the square root of a
      Negative number.");
      return Math.Ceiling (Math.Pow(number ,1.0/2.0));
    }
static void Main()
{
    console.Write ( "Enter a number to find its square root: ");
    double  sqrtInput= Convert.ToDouble ( Console.ReadLine());
    double sqrt Result= CalculateSquareRoot (sqrtInput);
    Console.Writeline( $"Square root of {sqrtInput} is
    approximatly: {sqrtResult}");
    Console.Write ("\n Enter a number to find its absolute
    value: ");
    double absInput= Convert.ToDouble (Console.Readline());
    double absResult= Find Absolute value (absInputl;
    Console.WriteLine ($" Absolute value of {absInput} is:
    {absRESult}");
    Console.ReadKey();
  }
}
```

Actual Output

Test Case1:

```
Enter a number to find its square root:16
Square root of 16 is approximatly: 4
Enter a number to find its absolute value: -9
Absolute value of -9 is: 9
```

1.10 Edges Cases

Code:

```
using System;
classEdge
{
 static void main()
 {

     double number1 = 0;
     double number2 = 0;
     double number3 = 0;
     double number4 = 0;

     number 1 = Math.Pow( double.Positive Infinity, 2);
     number 2 = Math.Pow( double.Negative Infinity, 2);
     number 3 = Math.Pow( double.MinValue, 0);
     number4 = Math.Pow (double.NaN, 2);

     Console.WriteLine ("Number1 : {0}", number1);
     Console.WriteLine ("Number2 : {0}", number2);
     Console.WriteLine ("Number3 : {0}", number 3);
     Console.WriteLine ("Number4 : {0}", number 4);
     Console.ReadKey();

  }
}
```

Actual Output

```
Number1 : ∞
Number 2 : ∞
Number 3 : 1
Number 4 : NaN
```