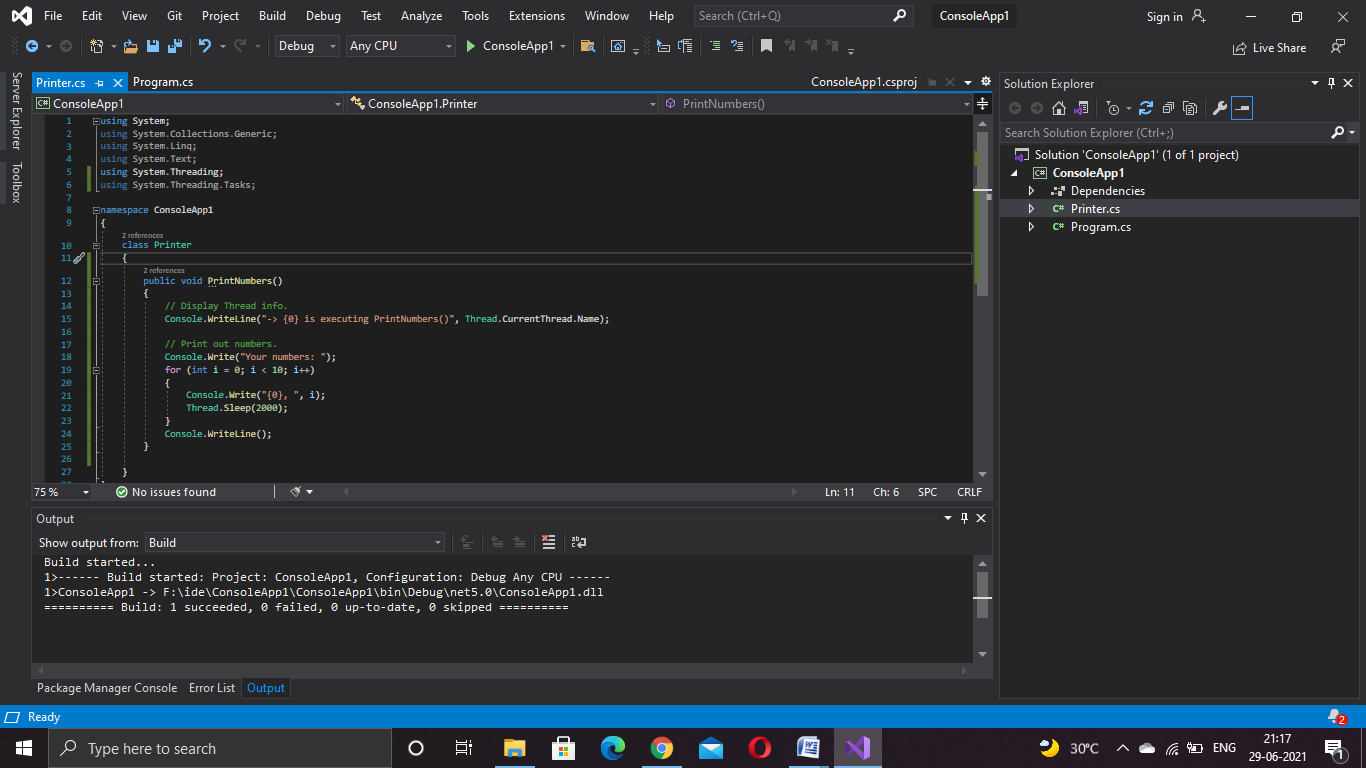
**DAY 3**

**C# Additional Topics Async Programming, Multi-threading**

**Multithreading - ThreadStart(Hands-On Lab)**

**Program:**

****

**Printer.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Printer

{

public void PrintNumbers()

{

// Display Thread info.

Console.WriteLine("-> {0} is executing PrintNumbers()", Thread.CurrentThread.Name);

// Print out numbers.

Console.Write("Your numbers: ");

for (int i = 0; i < 10; i++)

{

Console.Write("{0}, ", i);

Thread.Sleep(2000);

}

Console.WriteLine();

}

}

}

**Program.cs**

using System;

using System.Threading;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("\*\*\*\*\*ThreadStart Delegate App\*\*\*\*\*\n");

Console.Write("Do you want [1] or [2] threads? ");

string threadCount = Console.ReadLine();

// Name the current thread.

Thread primaryThread = Thread.CurrentThread;

primaryThread.Name = "Primary";

// Display Thread info.

Console.WriteLine("-> {0} is executing Main()", Thread.CurrentThread.Name);

// Make worker class.

Printer p = new Printer();

switch (threadCount)

{

case "2":

// Now make the thread.

Thread backgroundThread = new Thread(new ThreadStart(p.PrintNumbers));

backgroundThread.Name = "Secondary";

backgroundThread.Start();// Changes the state of current instance to ThreadState.Running.

break;

case "1":

p.PrintNumbers();

break;

default:

Console.WriteLine("I don't know what you want... you get 1 thread.");

goto case "1";

}

// Do some additional work.

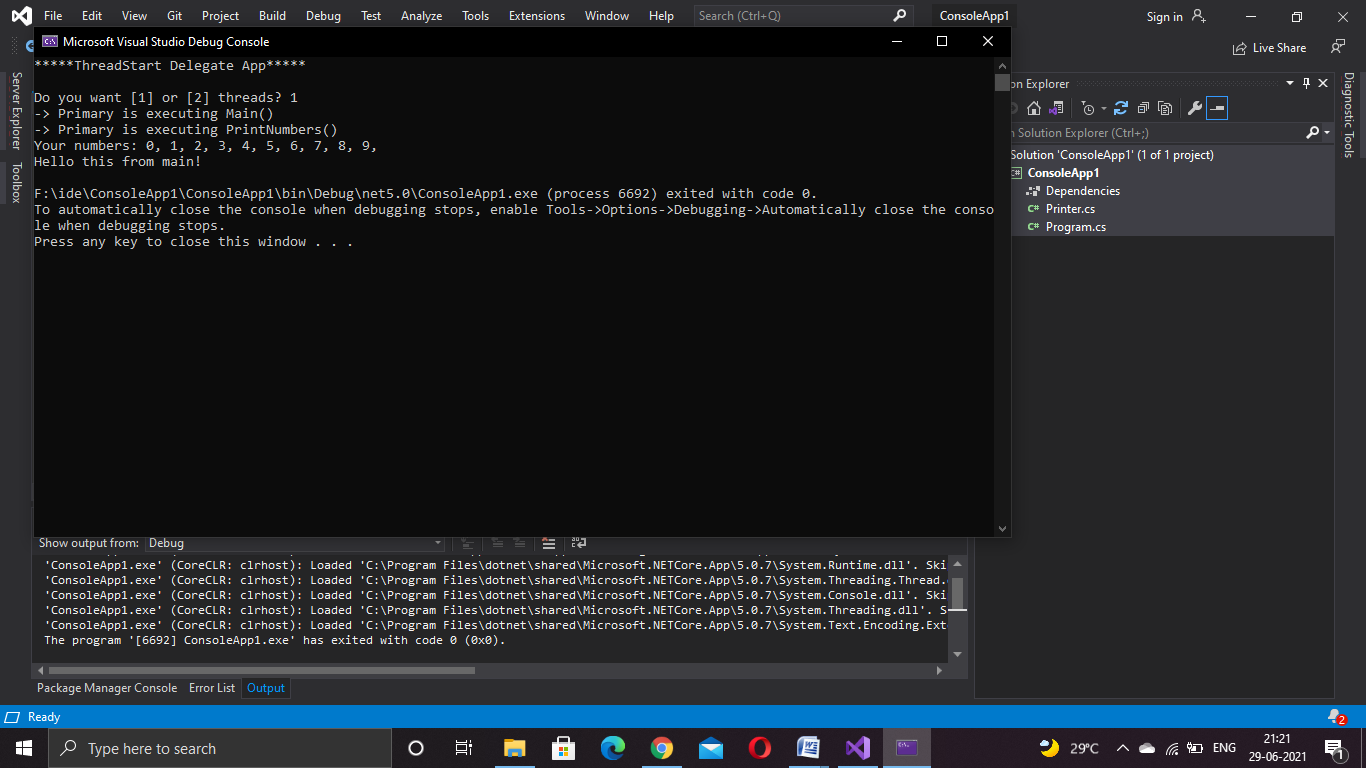
Console.WriteLine("Hello this from main!");

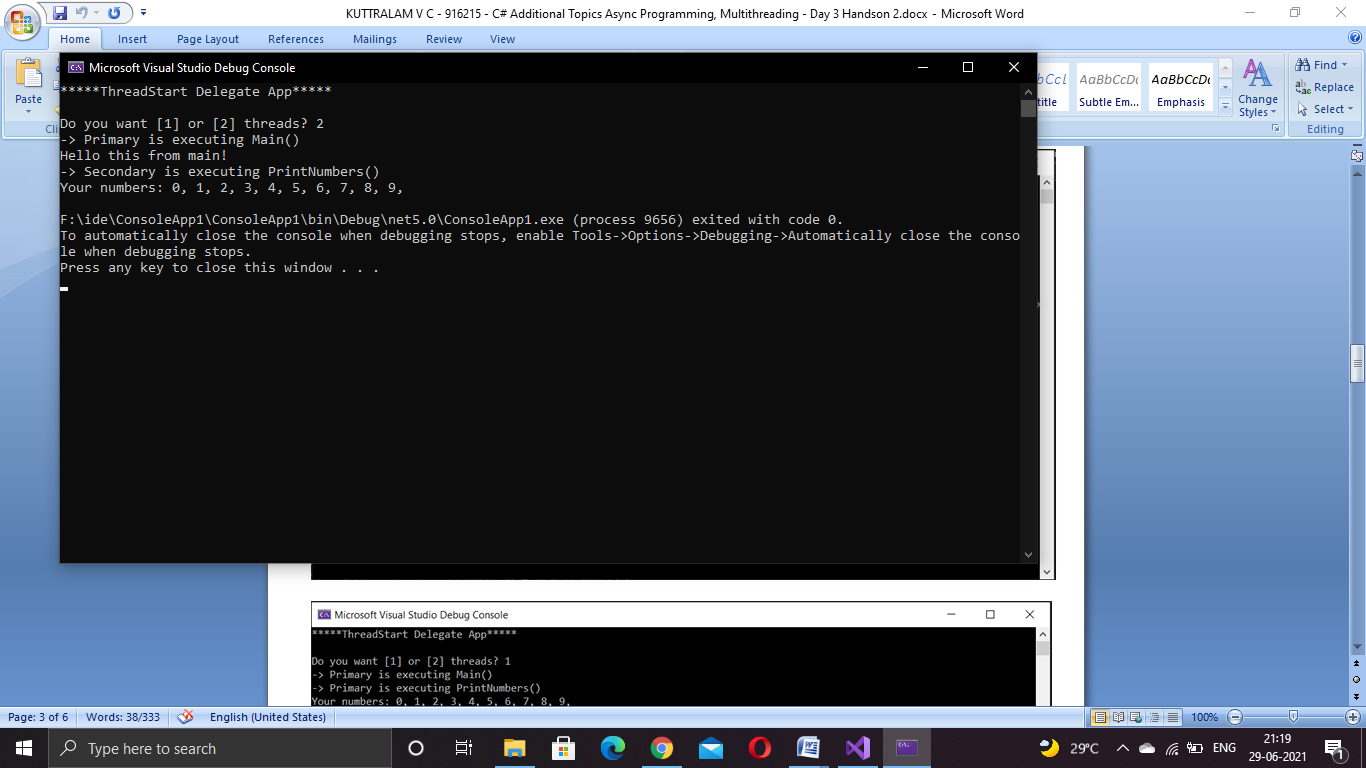
}

}

}

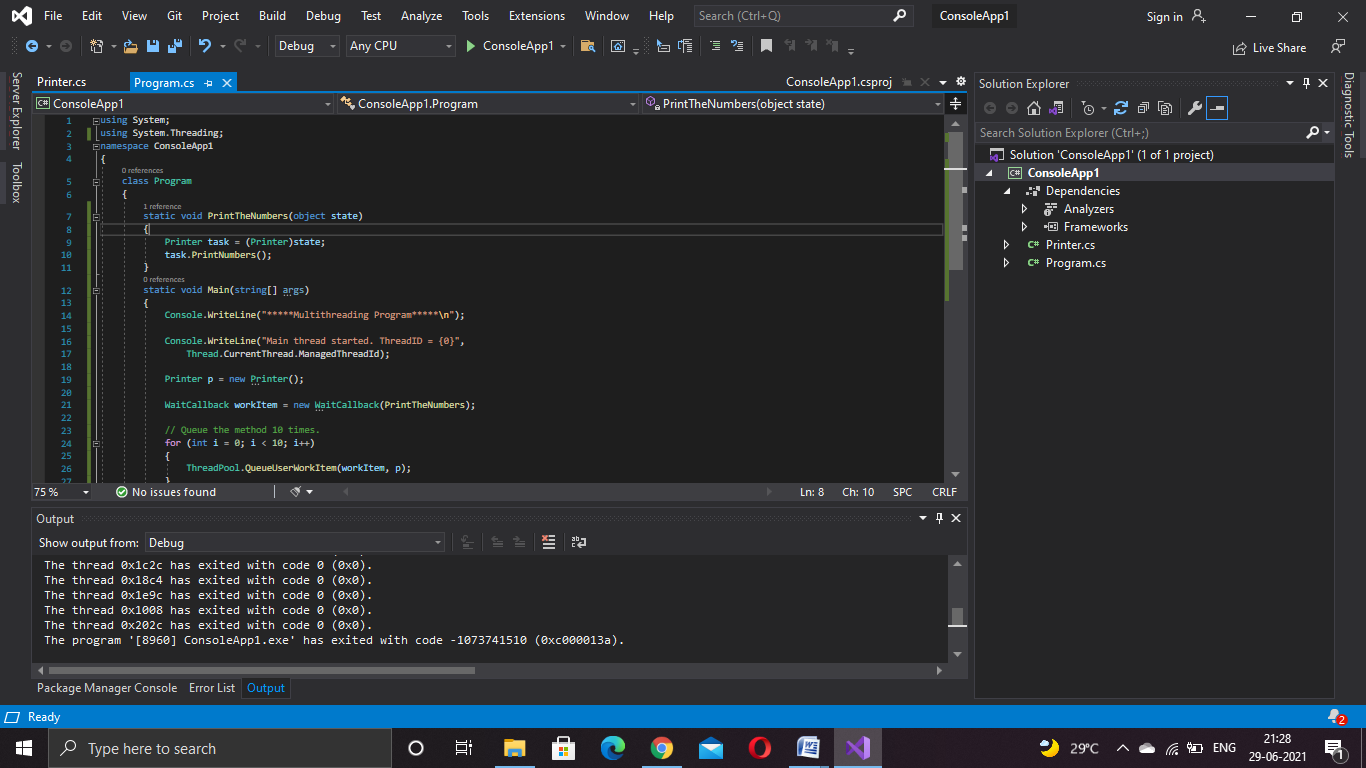
**Output:**

****

****

**Create a C# console application with the name “Multithreading Sample”**

**Program:**

****

**Printer.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Printer

{

private object lockToken = new object();

public void PrintNumbers()

{

lock (lockToken)

{

// Display Thread info.

Console.WriteLine("Thread-> {0} started @{1} and executing PrintNumbers() method",

Thread.CurrentThread.ManagedThreadId, DateTime.Now.ToLongTimeString());

// Print out numbers.

Console.Write("Your numbers: ");

for (int i = 0; i < 10; i++)

{

Console.Write("{0}, ", i);

Thread.Sleep(500);

}

Console.WriteLine();

}

}

}

}

**Program.cs**

using System;

using System.Threading;

namespace ConsoleApp1

{

class Program

{

static void PrintTheNumbers(object state)

{

Printer task = (Printer)state;

task.PrintNumbers();

}

static void Main(string[] args)

{

Console.WriteLine("\*\*\*\*\*Multithreading Program\*\*\*\*\*\n");

Console.WriteLine("Main thread started. ThreadID = {0}",

Thread.CurrentThread.ManagedThreadId);

Printer p = new Printer();

WaitCallback workItem = new WaitCallback(PrintTheNumbers);

// Queue the method 10 times.

for (int i = 0; i < 10; i++)

{

ThreadPool.QueueUserWorkItem(workItem, p);

}

Console.WriteLine("All task Queued");

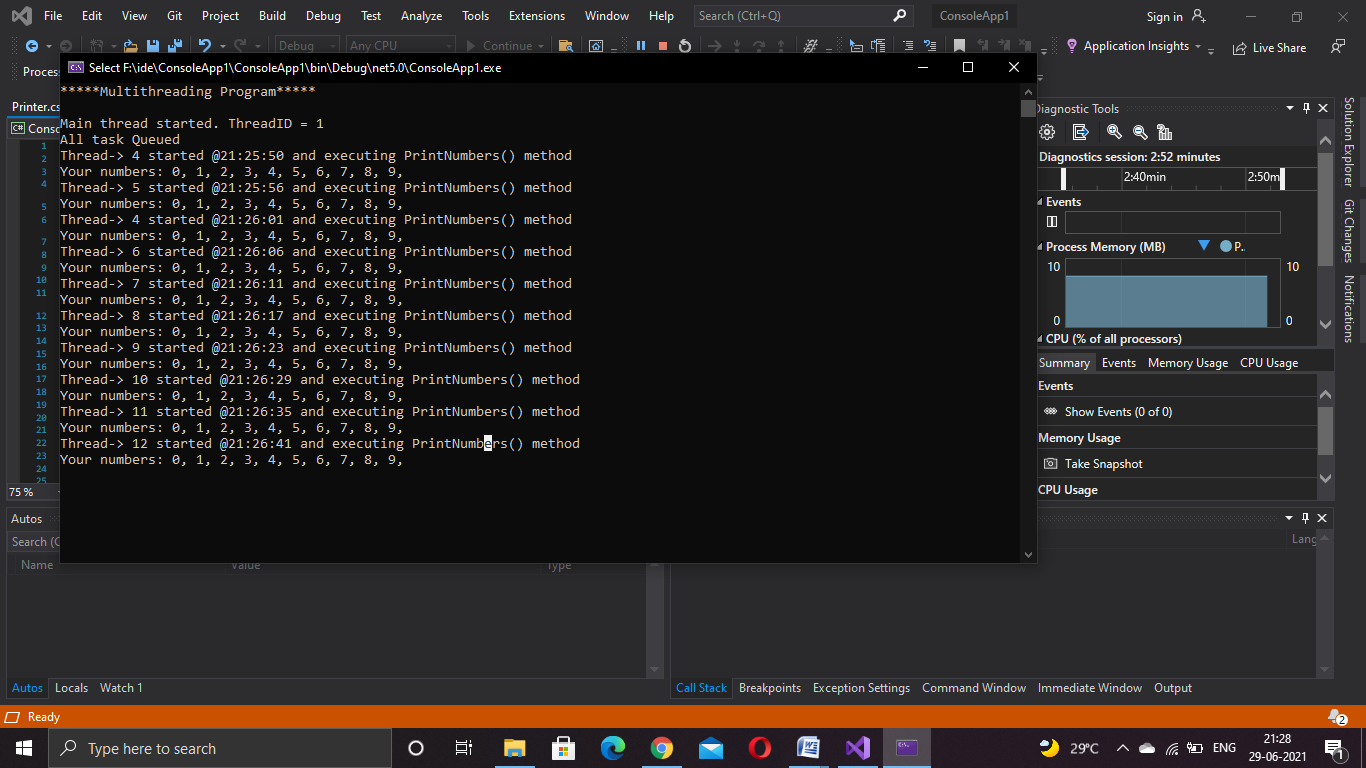
Console.ReadLine();

}

}

}

**Output:**

****