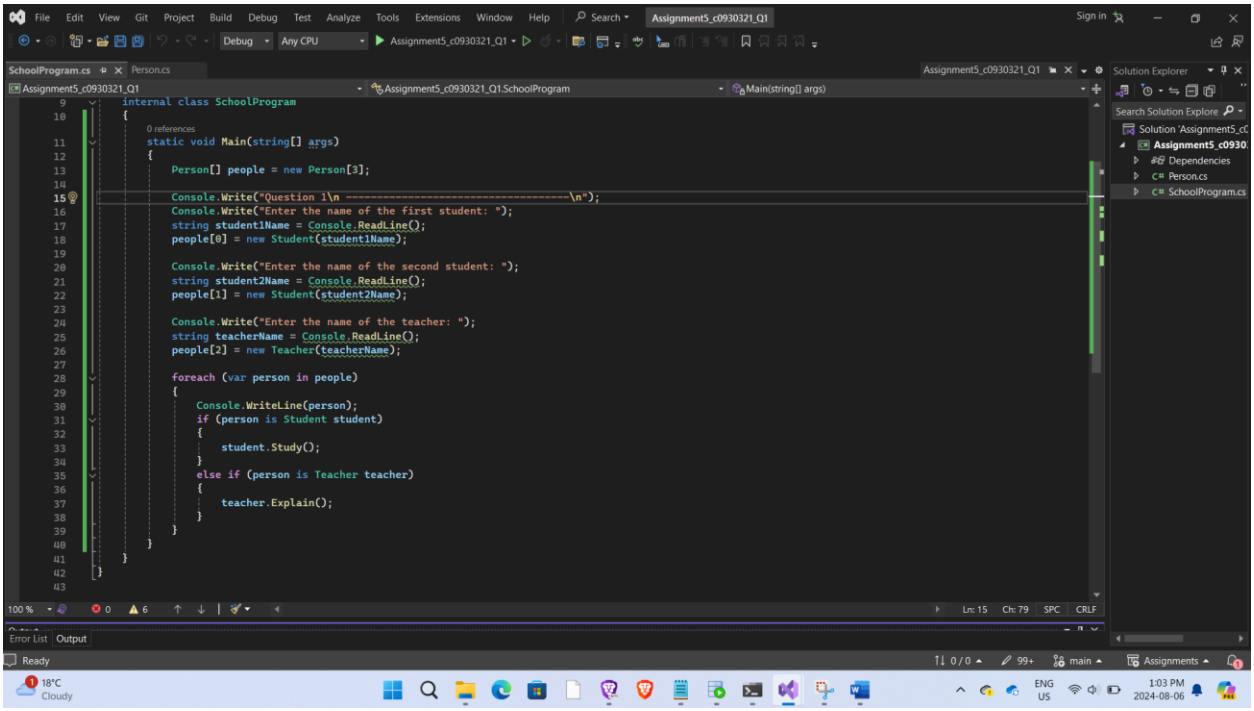
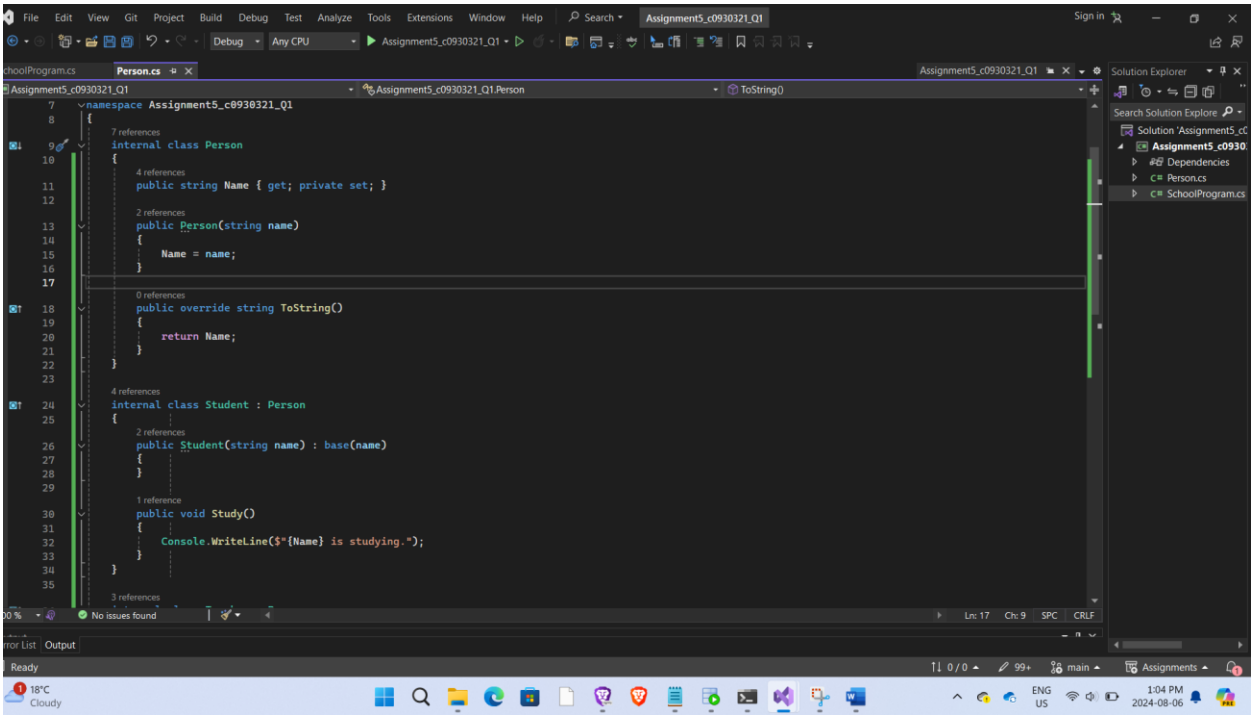
Question 1:

Code screenshot





Code

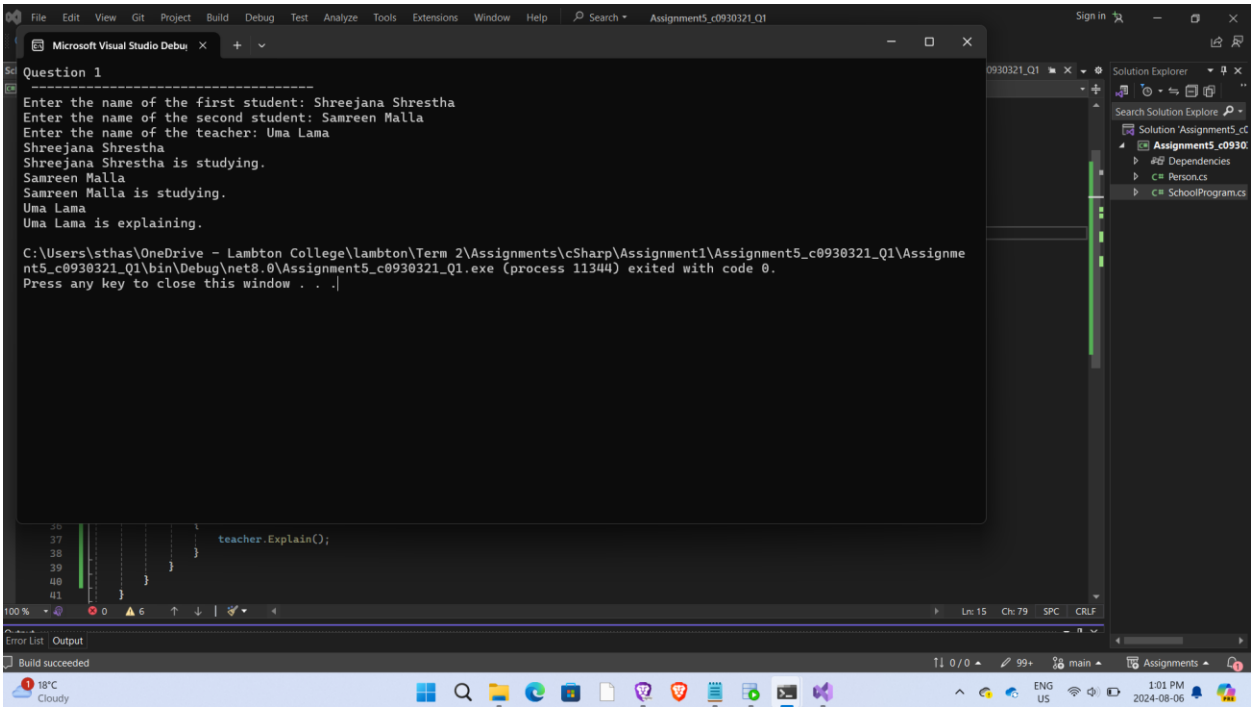
```
using System;



namespace Assignment5_c0930321_Q1
{
    internal class Person
    {
        public string Name { get; private set; }


        public Person(string name)
        {
            Name = name;
        }
```

```
    public override string ToString()
    {
        return Name;
    }
}

internal class Student : Person
{
    public Student(string name) : base(name)
    {
    }

    public void Study()
    {
        Console.WriteLine($"{Name} is studying.");
    }
}

internal class Teacher : Person
{
    public Teacher(string name) : base(name)
    {
    }

    public void Explain()
    {
        Console.WriteLine($"{Name} is explaining.");
```

```
        }
    }
}


using System;


namespace Assignment5_c0930321_Q1
{
    internal class SchoolProgram
    {
        static void Main(string[] args)
        {
            Person[] people = new Person[3];

            Console.Write("Question 1\n ----------------------------------\n");
            Console.Write("Enter the name of the first student: ");
            string student1Name = Console.ReadLine();
            people[0] = new Student(student1Name);

            Console.Write("Enter the name of the second student: ");
            string student2Name = Console.ReadLine();
            people[1] = new Student(student2Name);

            Console.Write("Enter the name of the teacher: ");
            string teacherName = Console.ReadLine();
```

```
        people[2] = new Teacher(teacherName);

        foreach (var person in people)
        {
            Console.WriteLine(person);
            if (person is Student student)
            {
                student.Study();
            }
            else if (person is Teacher teacher)
            {
                teacher.Explain();
            }
        }
    }
}
```

Output Screenshot



Question 2:

Code:

```
using System;


namespace Assignment5_Q2_c0930321
{
    public abstract class Shape
    {
        protected Location c;


        public Shape(Location c)
```

```
    {
        this.c = c;
    }

    public abstract double Area();
    public abstract double Perimeter();

    public override string ToString()
    {
        return $"Shape at location ({c.x}, {c.y})\n";
    }
}

public class Rectangle : Shape
{
    private double side1;
    private double side2;

    public Rectangle(Location c, double side1, double side2) : base(c)
    {
        this.side1 = side1;
        this.side2 = side2;
    }

    public override double Area()
    {
        return side1 * side2;
```

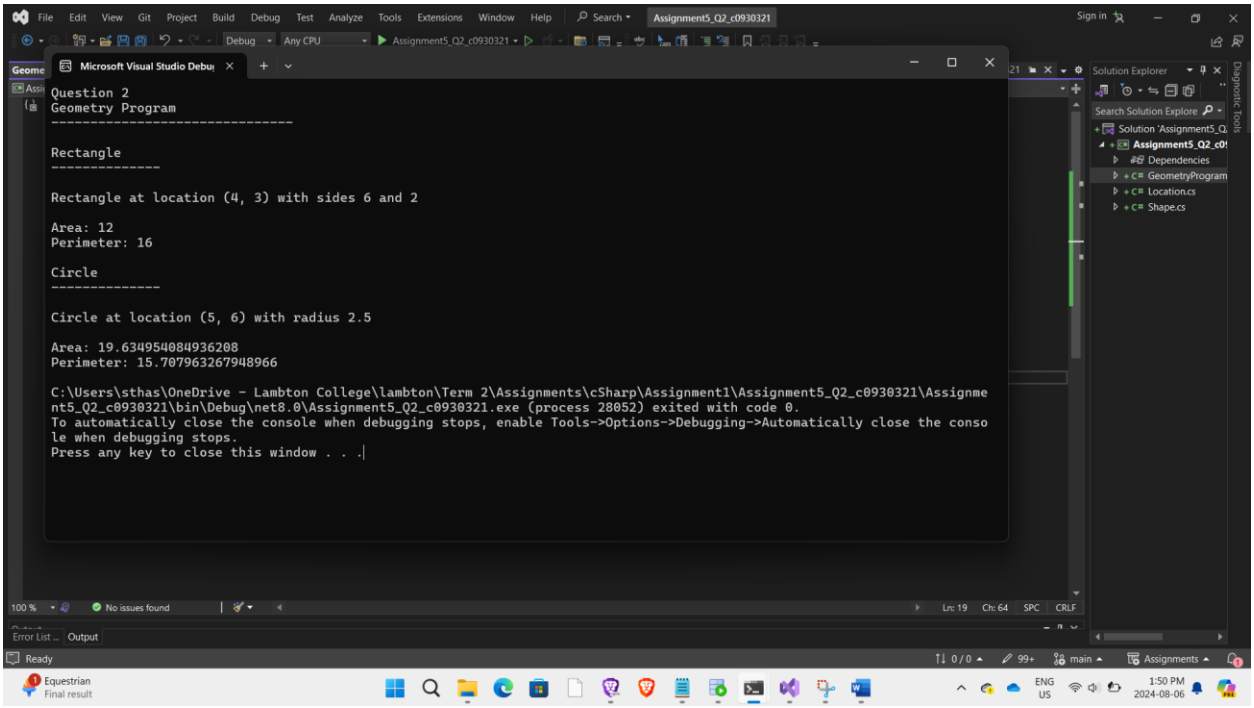
```
    }

    public override double Perimeter()
    {
        return 2 * (side1 + side2);
    }

    public override string ToString()
    {
        return $"Rectangle at location ({c.x}, {c.y}) with sides {side1} and {side2}\n";
    }
}

public class Circle : Shape
{
    private double radius;

    public Circle(Location c, double radius) : base(c)
    {
        this.radius = radius;
    }

    public override double Area()
    {
        return Math.PI * radius * radius;
    }
```

```
        public override double Perimeter()
        {
            return 2 * Math.PI * radius;
        }

        public override string ToString()
        {
            return $"Circle at location ({c.x}, {c.y}) with radius {radius}\n";
        }
    }
}


using System;

namespace Assignment5_Q2_c0930321
{
    public class Location
    {
        public double x { get; set; }
        public double y { get; set; }

        public Location(double x, double y)
        {
            this.x = x;
            this.y = y;
        }
```

```
    }
}

using System;

namespace Assignment5_Q2_c0930321
{
    internal class GeometryProgram
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Question    2\nGeometry    Program\n------------------------------
\n");
            Location location1 = new Location(4.0, 3.0);
            Console.WriteLine("Rectangle\n----------\n");
            Shape rect = new Rectangle(location1, 6.0, 2.0);
            Console.WriteLine(rect.ToString());
            Console.WriteLine($"Area: {rect.Area()}");
            Console.WriteLine($"Perimeter: {rect.Perimeter()}\n");

            Location location2 = new Location(5.0, 6.0);
            Console.WriteLine("Circle\n----------\n");
            Shape circle = new Circle(location2, 2.5);
            Console.WriteLine(circle.ToString());
            Console.WriteLine($"Area: {circle.Area()}");
            Console.WriteLine($"Perimeter: {circle.Perimeter()}");
        }
```

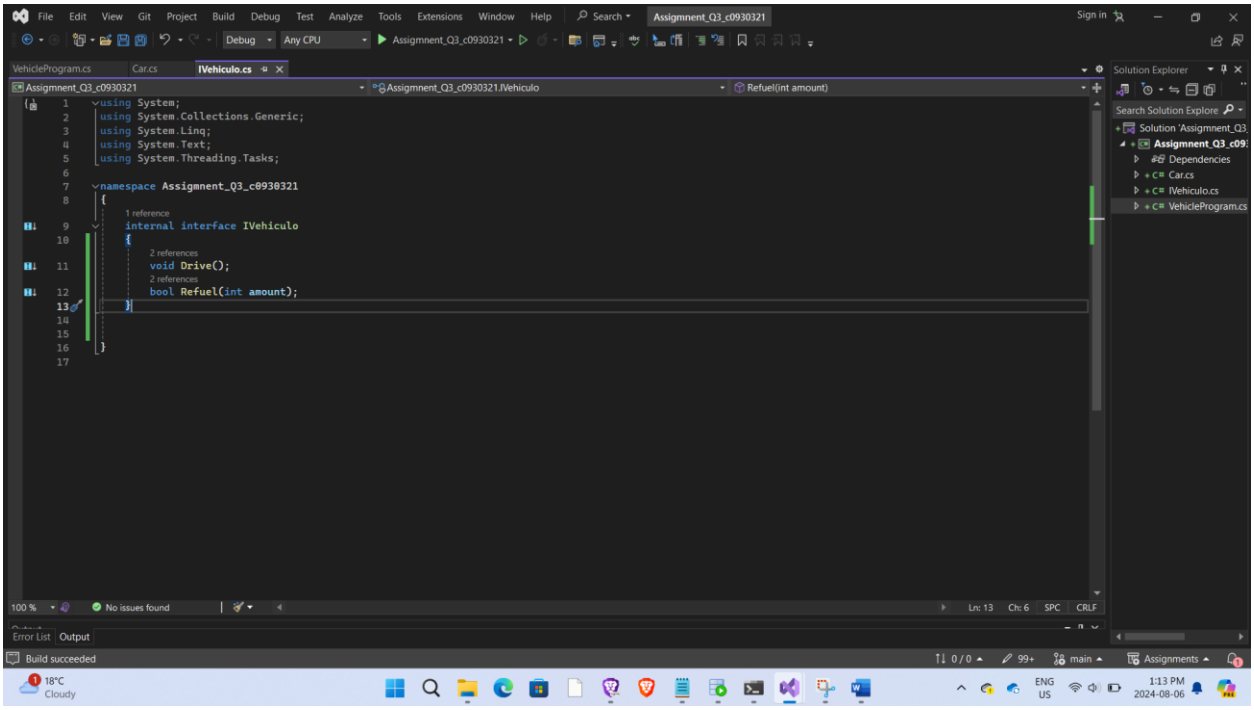
```
    }
}
```

Output Screenshot

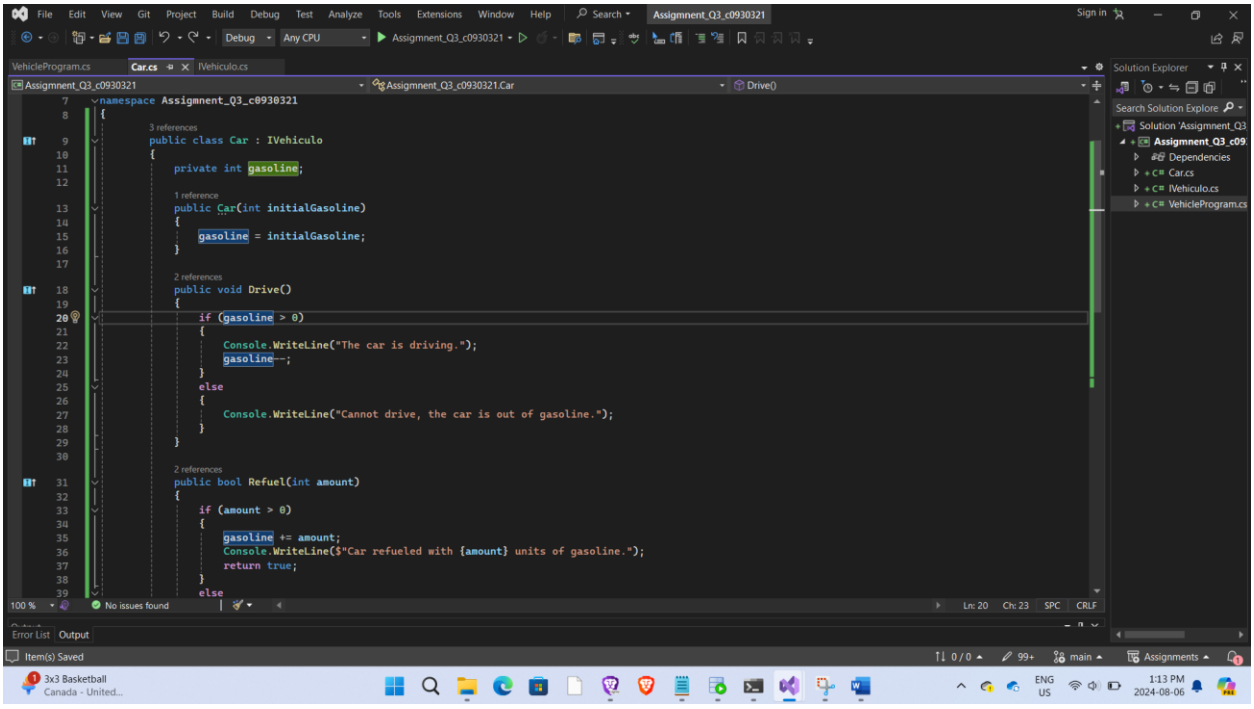


Question 3:

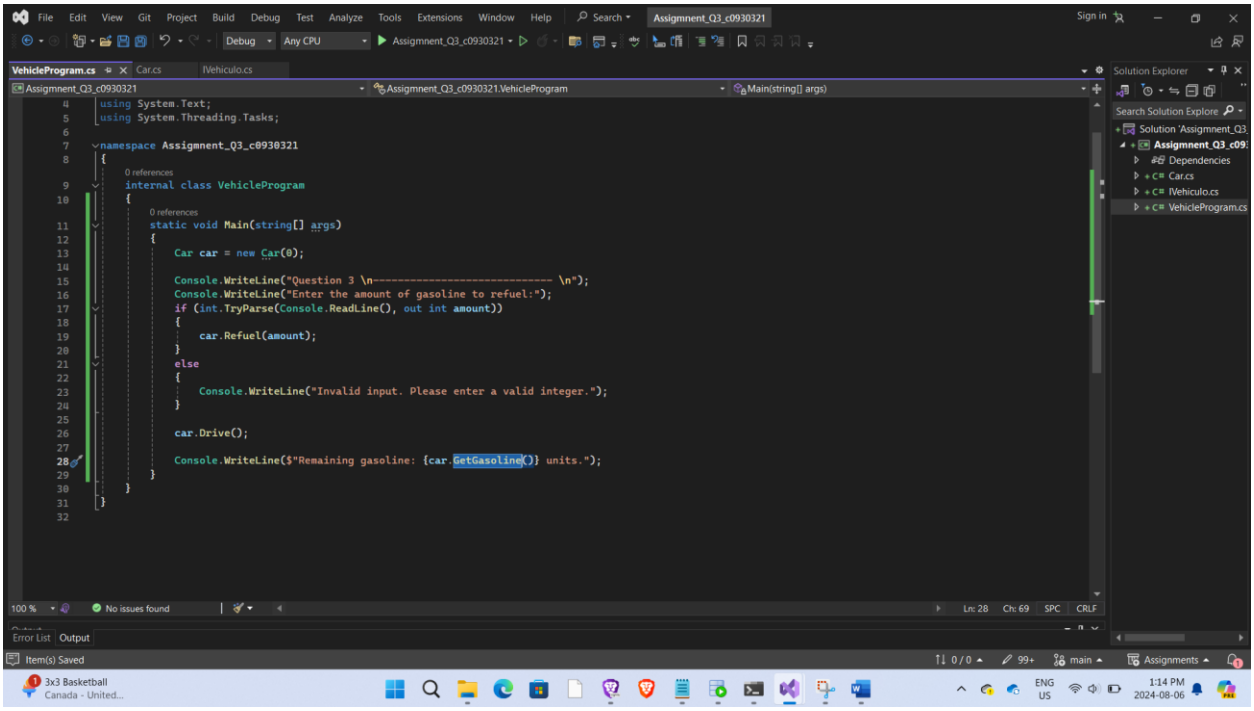
Code screenshot



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment_Q3_c0930321
{
    internal interface IVehiculo
    {
        void Drive();
        bool Refuel(int amount);
    }
}
```



```
namespace Assignment_Q3_c0930321
{
    public class Car : IVehiculo
    {
        private int gasoline;

        public Car(int initialGasoline)
        {
            gasoline = initialGasoline;
        }

        public void Drive()
        {
            if (gasoline > 0)
            {
                Console.WriteLine("The car is driving.");
                gasoline--;
            }
            else
            {
                Console.WriteLine("Cannot drive, the car is out of gasoline.");
            }
        }

        public bool Refuel(int amount)
        {
            if (amount > 0)
            {
                gasoline += amount;
                Console.WriteLine($"Car refueled with {amount} units of gasoline.");
                return true;
            }
            else
```



Code

```
using System;



namespace Assignment_Q3_c0930321
{
    internal class VehicleProgram
    {
        static void Main(string[] args)
        {
            Car car = new Car(0);

            Console.WriteLine("Question 3 \n---------- \n");
```

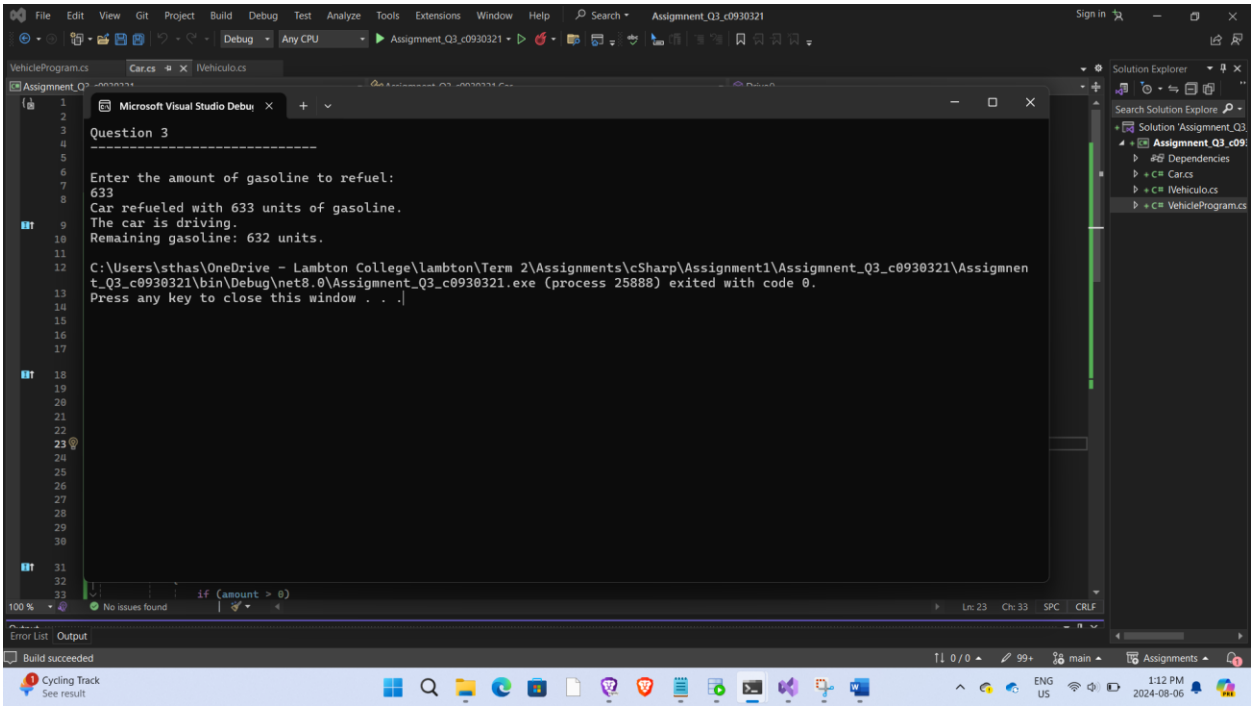
```
            Console.WriteLine("Enter the amount of gasoline to refuel:");
            if (int.TryParse(Console.ReadLine(), out int amount))
            {
                car.Refuel(amount);
            }
            else
            {
                Console.WriteLine("Invalid input. Please enter a valid integer.");
            }

            car.Drive();

            Console.WriteLine($"Remaining gasoline: {car.GetGasoline()} units.");
        }
    }
}



using System;



namespace Assigmnent_Q3_c0930321
{
        public class Car : IVehiculo
        {
            private int gasoline;
```

```
public Car(int initialGasoline)
{
    gasoline = initialGasoline;
}

public void Drive()
{
    if (gasoline > 0)
    {
        Console.WriteLine("The car is driving.");
        gasoline--;
    }
    else
    {
        Console.WriteLine("Cannot drive, the car is out of gasoline.");
    }
}

public bool Refuel(int amount)
{
    if (amount > 0)
    {
        gasoline += amount;
        Console.WriteLine($"Car refueled with {amount} units of gasoline.");
        return true;
    }
    else
```

```
            {
                Console.WriteLine("Invalid amount of gasoline to refuel.");
                return false;
            }
        }

        public int GetGasoline()
        {
            return gasoline;
        }
    }
}
```

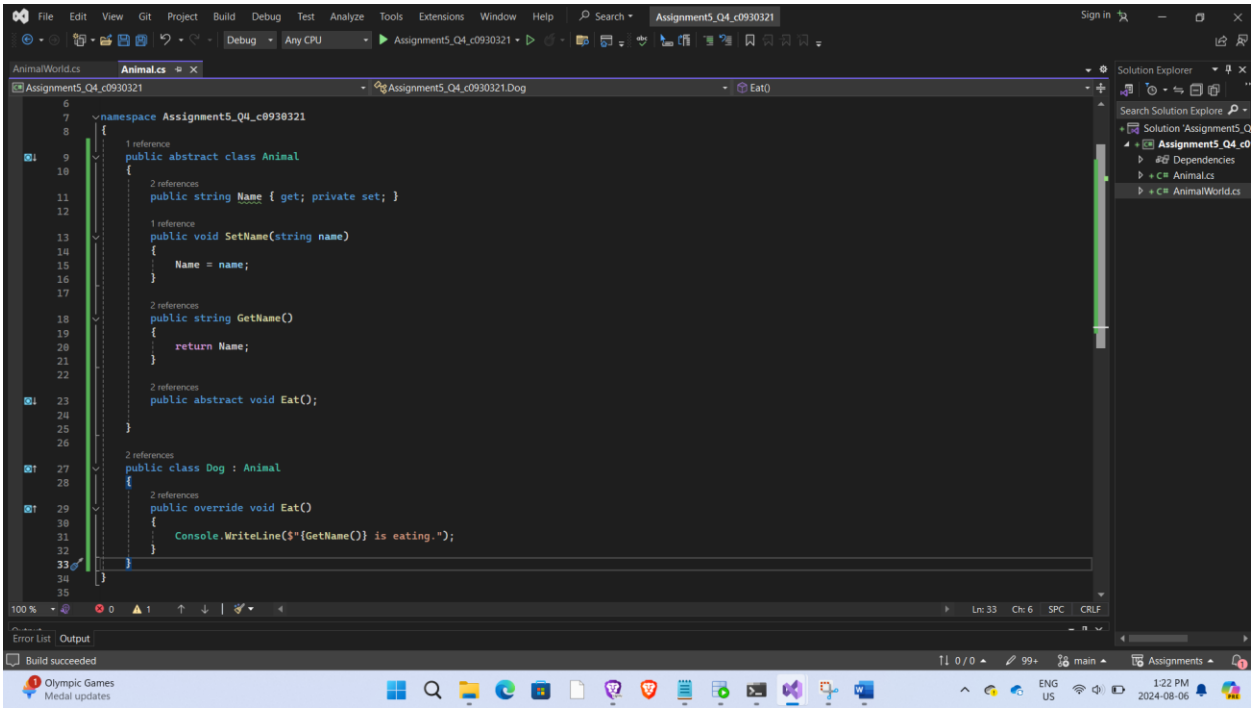
```
using System;


namespace Assigmnent_Q3_c0930321
{
    internal interface IVehiculo
    {
        void Drive();
        bool Refuel(int amount);
    }


}
```

Output Screenshot

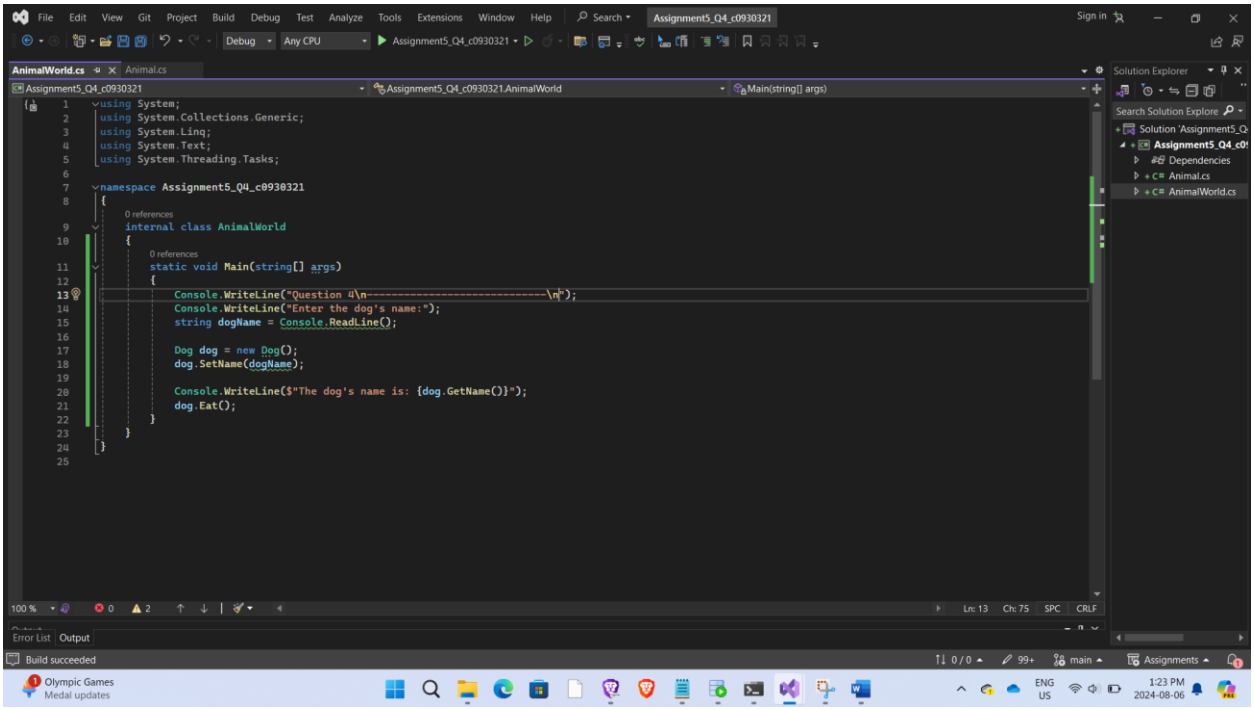


Question 4:

Code screenshot



```
namespace Assignment5_Q4_c0930321
{
    1 reference
    public abstract class Animal
    {
        2 references
        public string Name { get; private set; }

        1 reference
        public void SetName(string name)
        {
            Name = name;
        }

        2 references
        public string GetName()
        {
            return Name;
        }

        2 references
        public abstract void Eat();

    }

    2 references
    public class Dog : Animal
    {
        2 references
        public override void Eat()
        {
            Console.WriteLine($"{GetName()} is eating.");
        }
    }
}
```



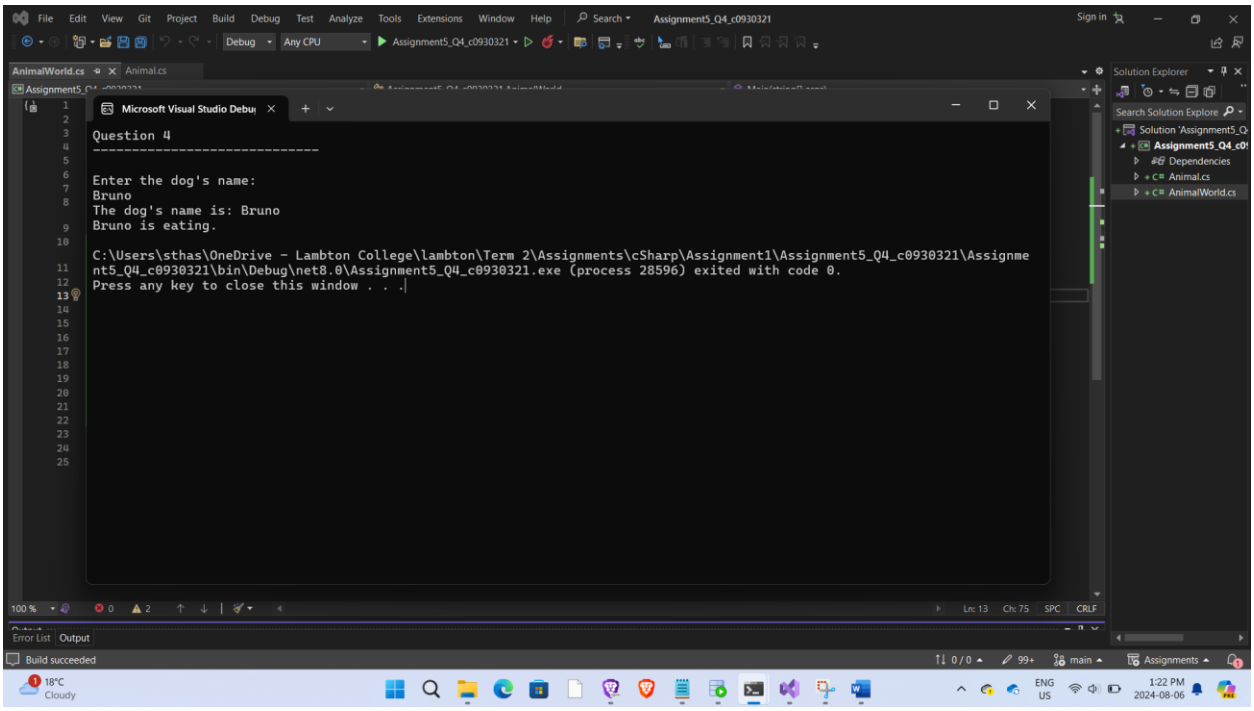
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment5_Q4_c0930321
{
    0 references
    internal class AnimalWorld
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Question 4\n----------------------------\n");
            Console.WriteLine("Enter the dog's name:");
            string dogName = Console.ReadLine();

            Dog dog = new Dog();
            dog.SetName(dogName);

            Console.WriteLine($"The dog's name is: {dog.GetName()}");
            dog.Eat();
        }
    }
}
```

Code

```
using System;



namespace Assignment5_Q4_c0930321
{
    internal class AnimalWorld
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Question 4\n----------------------------\n");
            Console.WriteLine("Enter the dog's name:");
            string dogName = Console.ReadLine();

            Dog dog = new Dog();
            dog.SetName(dogName);

            Console.WriteLine($"The dog's name is: {dog.GetName()}");
            dog.Eat();
        }
    }
}

using System;

namespace Assignment5_Q4_c0930321
{
```

```
    public abstract class Animal
    {
        public string Name { get; private set; }

        public void SetName(string name)
        {
            Name = name;
        }

        public string GetName()
        {
            return Name;
        }

        public abstract void Eat();

    }

    public class Dog : Animal
    {
        public override void Eat()
        {
            Console.WriteLine($"{GetName()} is eating.");
        }
    }
}
```

Output Screenshot



Question 5:

Code:

```
using System;


namespace Assignment5_Q5_c0930321
{
    internal class Student
    {
        private static double serviceTax = 12.3;
        public static double ServiceTax
        {
            get { return serviceTax; }
```

```
    set { serviceTax = value; }
}

public int RollNo { get; private set; }
public string Name { get; private set; }
public Course CourseJoined { get; private set; }
public double FeePaid { get; private set; }

public Student(int rollNo, string name, Course course)
{
    RollNo = rollNo;
    Name = name;
    CourseJoined = course;
    FeePaid = 0;
}

public void Payment(double amount)
{
    FeePaid += amount;
}

public double DueAmount
{
    get { return TotalFee - FeePaid; }
}

public double TotalFee
```

```
        {
            get { return CourseJoined.GetTotalFee() * (1 + ServiceTax / 100); }
        }

        public void Print()
        {
            Console.WriteLine($"Roll No: {RollNo}\n, Name: {Name}\n, Course: {CourseJoined.Name}\n, Fee Paid: {FeePaid:C}\n, Due Amount: {DueAmount:C}\n");
        }
    }
}

using System;

namespace Assignment5_Q5_c0930321
{
    public abstract class Course
    {
        public string Name { get; private set; }
        public int Duration { get; private set; }
        public double CourseFee { get; private set; }

        public Course(string name, int duration, double courseFee)
        {
            Name = name;
            Duration = duration;
            CourseFee = courseFee;
```

```
        }

        public abstract double GetTotalFee();

        public virtual void Print()
        {
            Console.WriteLine($"Course: {Name}, Duration: {Duration} months, Course Fee: {CourseFee:C}\n");
        }
    }

    class PartTimeCourse : Course
    {
        public string Timing { get; private set; }

        public PartTimeCourse(string name, int duration, double courseFee, string timing)
            : base(name, duration, courseFee)
        {
            Timing = timing;
        }

        public override double GetTotalFee()
        {
            return CourseFee * 0.9;
        }

        public override void Print()
```

```
    {
        base.Print();
        Console.WriteLine($"Timing: {Timing}");
    }
}

class OnsiteCourse : Course
{
    public string CompanyName { get; private set; }
    public int NumberOfCandidates { get; private set; }

    public OnsiteCourse(string name, int duration, double courseFee, string companyName, int numberOfCandidates)
        : base(name, duration, courseFee)
    {
        CompanyName = companyName;
        NumberOfCandidates = numberOfCandidates;
    }

    public override double GetTotalFee()
    {
        return CourseFee * 1.1;
    }

    public override void Print()
    {
        base.Print();
```

```
        Console.WriteLine($"Company: {CompanyName},\n Number of Candidates: {NumberOfCandidates}\n");
    }

  }
}

using System;
namespace Assignment5_Q5_c0930321
{
  internal class StudentCourseMgmt
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Question 5\n Student Course Management\n----------\n");
      // Creating courses
      Course csharpCourse = new PartTimeCourse("C#", 6, 2000, "Evening");
      Course aspNetCourse = new OnsiteCourse("ASP.NET", 4, 3000, "ABC Corp", 10);
      Course pythonCourse = new PartTimeCourse("Python", 5, 1500, "Morning");

      // Creating students
      Student student1 = new Student(1, "Alish", csharpCourse);
      Student student2 = new Student(2, "Alishma", aspNetCourse);
      Student student3 = new Student(3, "Karishma", pythonCourse);

      // Simulate payments flow
```

```
            student1.Payment(500);
            student2.Payment(1000);
            student3.Payment(1500);

            // Printing student details
            student1.Print();
            student2.Print();
            student3.Print();

            // Printing course details
            csharpCourse.Print();
            aspNetCourse.Print();
            pythonCourse.Print();
        }
    }
}
```

Output Screenshot: