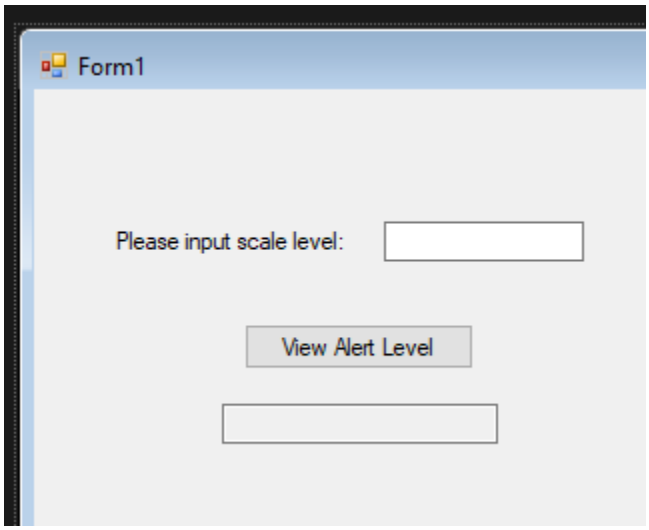
File name: CIS199E1S22

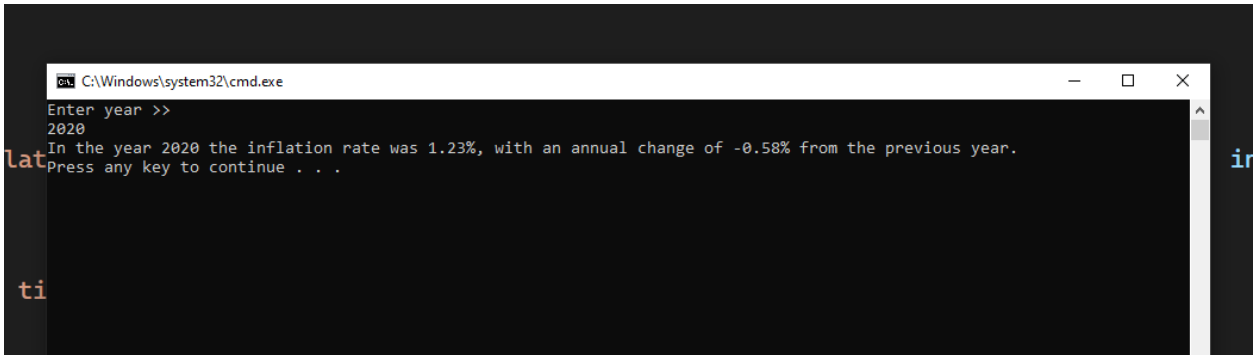


In Form1.cs:

```
/* Exam 1 Part 3
   Grading ID: K3215
   Due Date: 2/16/2022
   Comment: Form to display alert level.
 */

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CIS199E1S22
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        // button to display alert level
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            // defining the variables
            int scale;
            string displayScale;

            // reads string input and converts into integer.
            scale = int.Parse(inputScale.Text);

            // condition statements for alert levels.
            if (scale == 0)
                displayScale = "At rest";
            else if (scale <= 2)
                displayScale = "Active";
            else if (scale > 2 && scale <= 5)
                displayScale = "Imminent";
            else
                displayScale = "Invalid";

            // display the alert level to textbox.
            displayText.Text = String.Format("{0}", displayScale);
        }
    }
}
```

File name: Exam2_S22



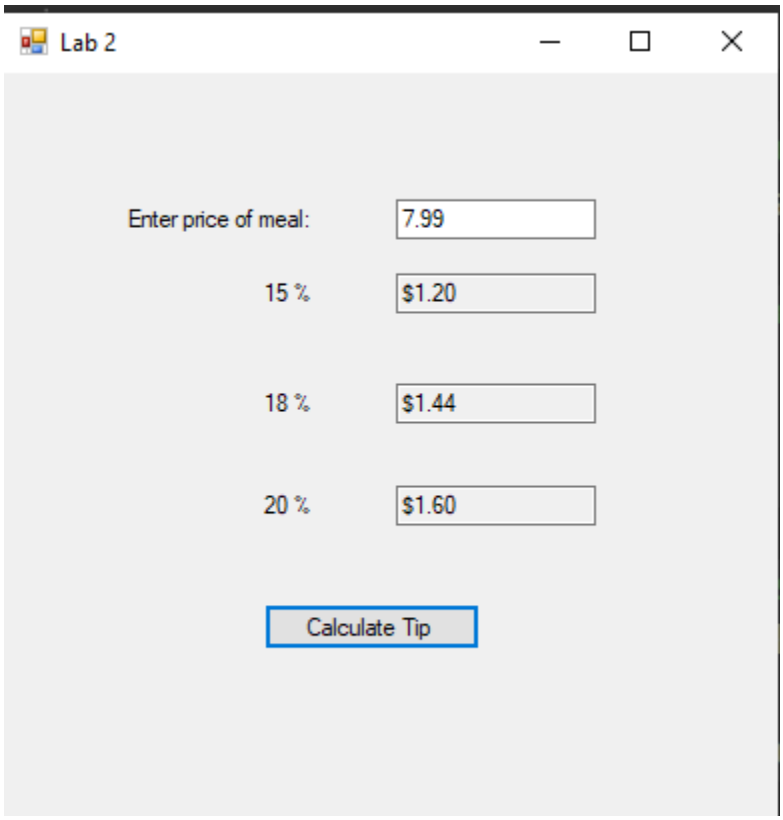
In Program.cs:

```
//Grading ID K3215
//Console app to match year with inflation rates and annual changes.
```

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

namespace Exam2_S22
{
   class Program
   {
      static void Main(string[] args)
      {
         int[] years = { 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020 };
         double[] inflationRates = { .0164, .0316, .0207, .0146, .0162, .0012, .0126, .0213, .0244,
.0181, .0123 };
         double[] annualChanges = { .02, .0152, -.0109, -.0060, .0016, -.0150, .0144, .0087,
.0031, -.0063, -.0058 };
         bool found = false;
         double foundRates = 0;
         double foundChanges = 0;
         int input;

         Console.WriteLine("Enter year >> ");

         input = int.Parse(Console.ReadLine());

         for (int i=0; i<years.Length && !found; i++)
         {
            if ( input == years[i] )
            {
               found = true;
               foundRates = inflationRates[i]*100;
               foundChanges = annualChanges[i]*100;
               break;
            }
         }

         if (found == true)
         {
            Console.WriteLine("In the year {0} the inflation rate was {1:N2}%, with an annual
change of {2:N2}% from the previous year.", input, foundRates, foundChanges);
         }
         else if (!found)
         {
```

```
            Console.WriteLine("Enter an int value next time.");

            //was unable to finish the error messages

        }
    }
    }
}
```

File name: Lab2

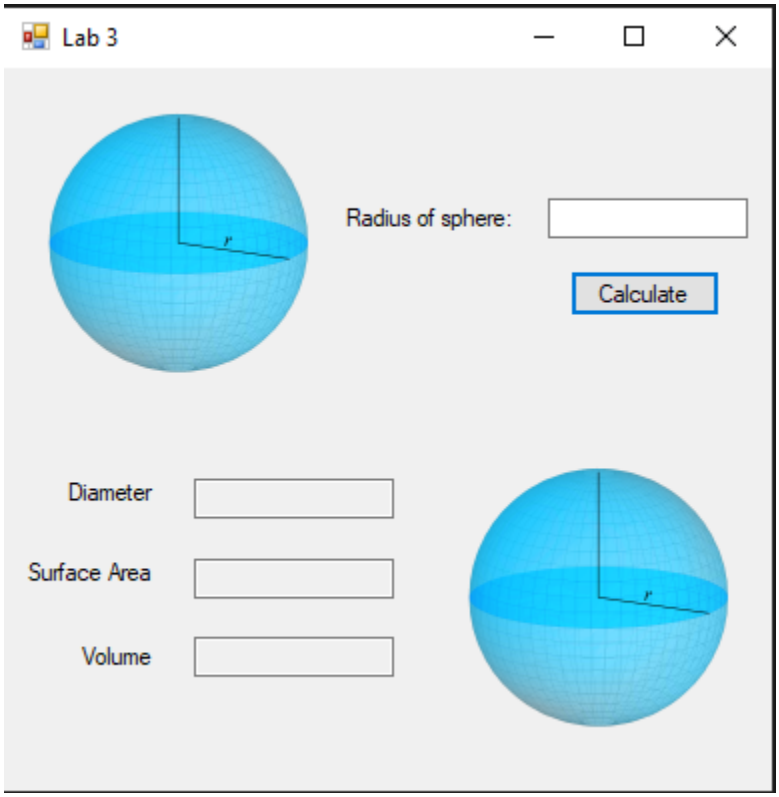


Form1.cs:

```
/*Lab 2
  Grading ID: K3215
  Course section: CIS 199-01
  Due date: 2/6/2022
  Description: Form to calculate tip rate according to the numeric input.
  */
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lab2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // defining constant tip rates 15, 18, and 20 percent respectively.
            const double TIPRATE_ONE = 0.15;
            const double TIPRATE_TWO = 0.18;
            const double TIPRATE_THREE = 0.20;

            // convert user input to double.
            double inputVal = double.Parse(inputValue.Text);

            // calculating the tip rates using the input value and the named constants.
            double valOne = inputVal * TIPRATE_ONE;
            double valTwo = inputVal * TIPRATE_TWO;
            double valThree = inputVal * TIPRATE_THREE;

            // display result to corresponding textbox, in dollar format with 2 decimals.
            calcFifteen.Text = String.Format("{0}", valOne.ToString("C2"));
            calcEighteen.Text = String.Format("{0}", valTwo.ToString("C2"));
            calcTwenty.Text = String.Format("{0}", valThree.ToString("C2"));

        }
    }
}
```

File name: Lab3

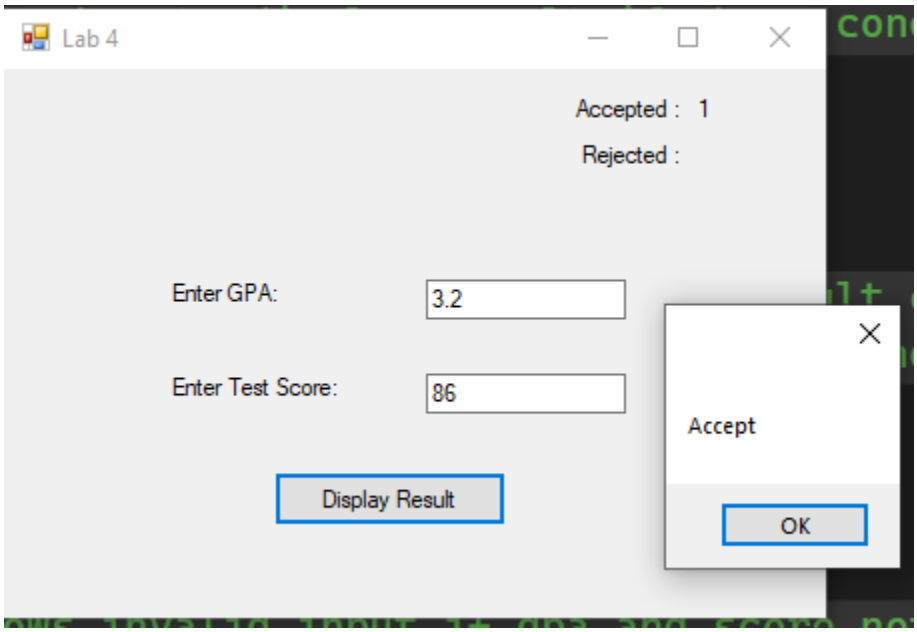


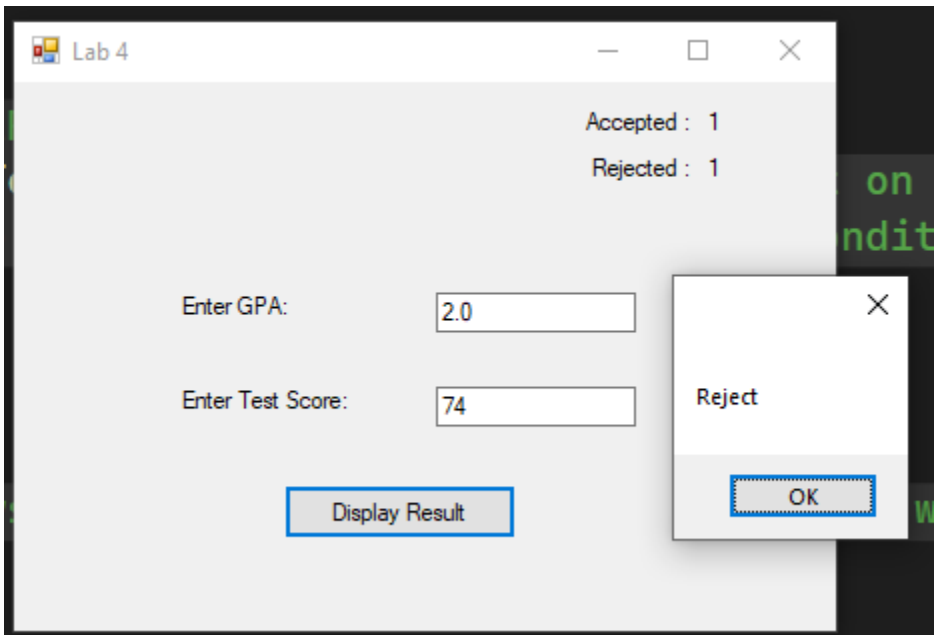
Form1.cs:

```
/* Lab 3
   Grading ID: K3215
   Due date: 2/13/22
   Course section: CIS 199-01
   Comment: Form that calculates and displays several geometric values using a radius from user input.
 */

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace Lab3
{
   public partial class Lab3 : Form
   {
      public Lab3()
      {
         InitializeComponent();
      }

      // button to calculate diameter, surface area, and volume of input radius with given
formulas.
      private void calculateButton_Click(object sender, EventArgs e)
      {
         //
         double diameter;
         double surfaceArea;
         double volume;
         double radius;

         //convert radius input to double.
         radius = double.Parse(radiusText.Text);

         //calculate diameter, surface area, and volume.
         diameter = 2 * radius;
         surfaceArea = 4 * Math.PI * Math.Pow(radius,2);
         volume = (4 * Math.PI * Math.Pow(radius, 3))/3;

         //display diameter, surface area, and volume in text box.
         diameterText.Text = String.Format("{0}", diameter.ToString("N2"));
         surfaceText.Text = String.Format("{0}", surfaceArea.ToString("N2"));
         volumeText.Text = String.Format("{0}", volume.ToString("N2"));
      }
   }
}
```

File name: Lab4





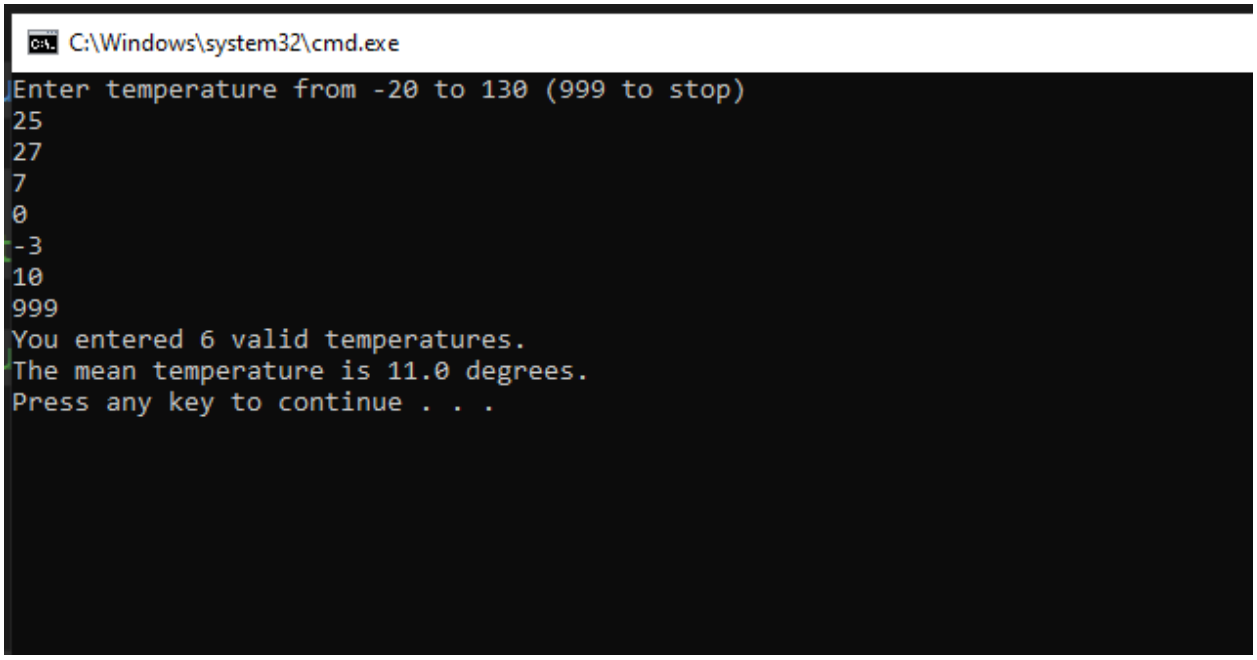
Form1.cs:

```
/*Lab 4
  Grading ID: K3215
  Due date: 2/27/22
  Course section: CIS 199-01
  Comment: Form to determine whether gpa and score of student is accepted or rejected.
*/
using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lab4
{
   public partial class Lab4 : Form
   {
      // defining the variables for total accepted and rejected form entries
       int totalAccept = 0;
       int totalReject = 0;

       public Lab4()
       {
           InitializeComponent();
       }

       // button to display results of form
       private void button1_Click(object sender, EventArgs e)
       {
           // defining variables gpa and score
           float gpa;
           int score;


           if (float.TryParse(gpaText.Text, out gpa) && int.TryParse(scoreText.Text, out score))
//gpa and score from user input, tryparse determines if variables are within the given scope
           {
               if (gpa >= 0.0 && gpa <= 4.0 && score >= 0 && score <= 100) //logical statements
defining the range of gpa and score
               {
                   if ((gpa >= 3.0 && score >= 60) || (gpa < 3.0 && score >= 80)) //logical statements
defining the conditions for an accepted entry
                   {
                       totalAccept++; //adding one to totalAccept
                       acceptTotalLabel.Text = totalAccept.ToString(); //displaying the result on label
after one is added to totalReject
                       MessageBox.Show("Accept"); // message box to display result if above
conditions are true
```

```
            }
            else
            {
                totalReject++; // adding one to totalReject
                rejectTotalLabel.Text = totalReject.ToString(); //displaying the result on label after
one is added to totalReject
                MessageBox.Show("Reject"); // message box to display result if above conditions
are false
            }
        }
        else
        {
            MessageBox.Show("Invalid input."); //shows invalid input if gpa and score not within
range
        }
    }
    else
        MessageBox.Show("Invalid input."); //shows invalid input if tryparse does not succeed



  }
 }
}
```

File name: Lab5

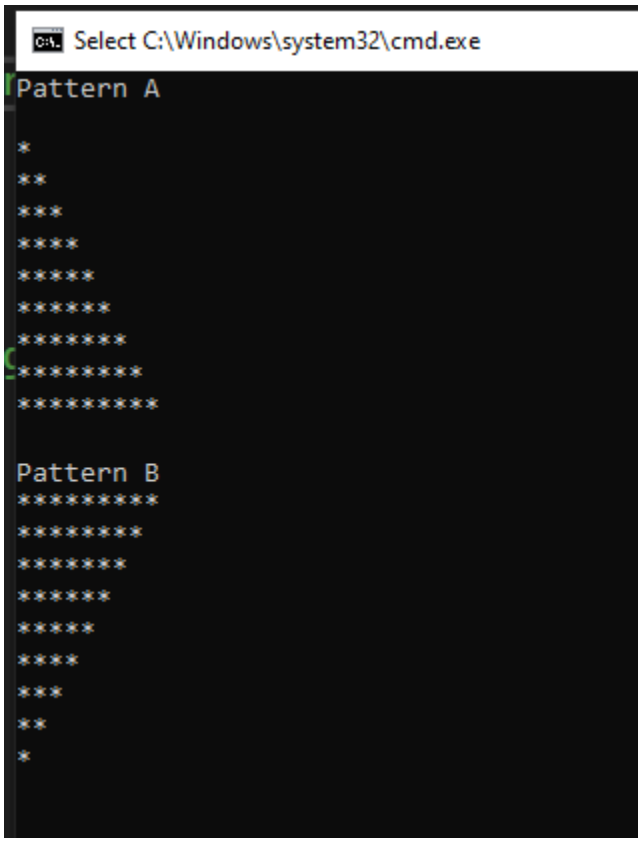


Program.cs:

```
/*Lab 5
 Course: CIS199-01
 Grading ID: K3215
 Due date: 3/6/22
 Comment: Console program to find the mean of input temperature values.
 */
using System;
using static System.Console;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab5
{
    class Program
    {
        static void Main(string[] args)
        {
            //defining variables
            int temp = 0;
            int init = 0;
            double sum = 0;
            double mean = 0;

            //promts user to enter value
            WriteLine("Enter temperature from -20 to 130 (999 to stop)");

            //loop while value is not 999
            while (temp != 999)
            {
                if (int.TryParse(ReadLine(), out temp) && temp >= -20 && temp <= 130 && temp !=
999) //logic statement for if value entered is an integer, within the -20 to 130 range, and not 999
                {
                    init++; // adding to total count
                    sum += temp; // sum for total temperature
                }
                else if (temp == 999) //if value entered is 999, break to jump out of loop
                {
                    break;
                }
```

```
            else
            {
                WriteLine("Valid temperatures range from -20 to 130. Please reenter
temperature."); //prompt user to reenter value due to invalid number from tryparse
            }
        }


        mean = sum / init; //finding the mean, total temp divided by valid values
        WriteLine("You entered {0} valid temperatures.", init); // display number of valid input
        WriteLine("The mean temperature is {0:N1} degrees.", mean); // display mean
temperature with 1 decimal point

    }
  }
}
```

File name: Lab6



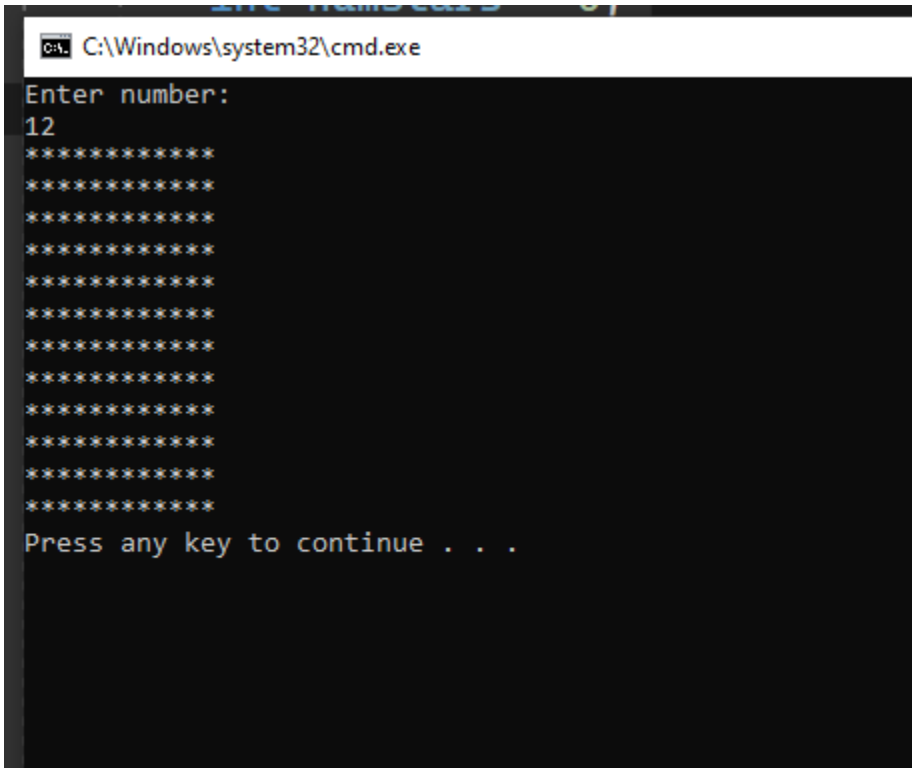
Program.cs:

```
/*Lab 6
  Grading ID: K3215
  Due date: 3/13/22
  Course: CIS 199-01
  Comment: Console application to print stars in certain patterns.
*/

using System;
using static System.Console;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab6
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            //variables for the rows
            int numA = 10;
            int numB = 1;

            WriteLine("Pattern A");
            for (int i = 1; i <= numA; i++) //number of rows to print or loop
            {
                for (int j = 1; j < i; j++ ) //for printing number of stars each row
                {
                    Write("*");
                }
                WriteLine();
            }

            WriteLine();
            WriteLine("Pattern B");
            for (int i = 10; i >= numB; i--) //for number of rows
            {
                for (int j = 1; j < i; j++) //for printing number of stars
                {
                    Write("*");
                }
                WriteLine();
            }

            WriteLine();
            WriteLine("Pattern C");
            for (int i = 1; i <= numA; i++) //for number of rows
            {

                for (int k = 1; k < i; k++) //printing number of spaces for each row
                {
                    Write(" ");

                }
                for (int j = 10; j > i; j--) //printing asterisk at a decreasing amount each row
                {
                    Write("*");
                }
```

```
            WriteLine();
        }

        WriteLine();
        WriteLine("Pattern D");
        for (int i = 1; i <= numA; i++) //for number of rows
        {

            for (int k = 9; k >= i; k--) //printing spaces at a decreasing amount in each row
            {
                Write(" ");

            }
            for (int j = 1; j < i; j++) //printing asterisk at an increasing amount each time it is looped
            {
                Write("*");
            }
            WriteLine();
        }
    }
  }
}
```

File name: Lab7

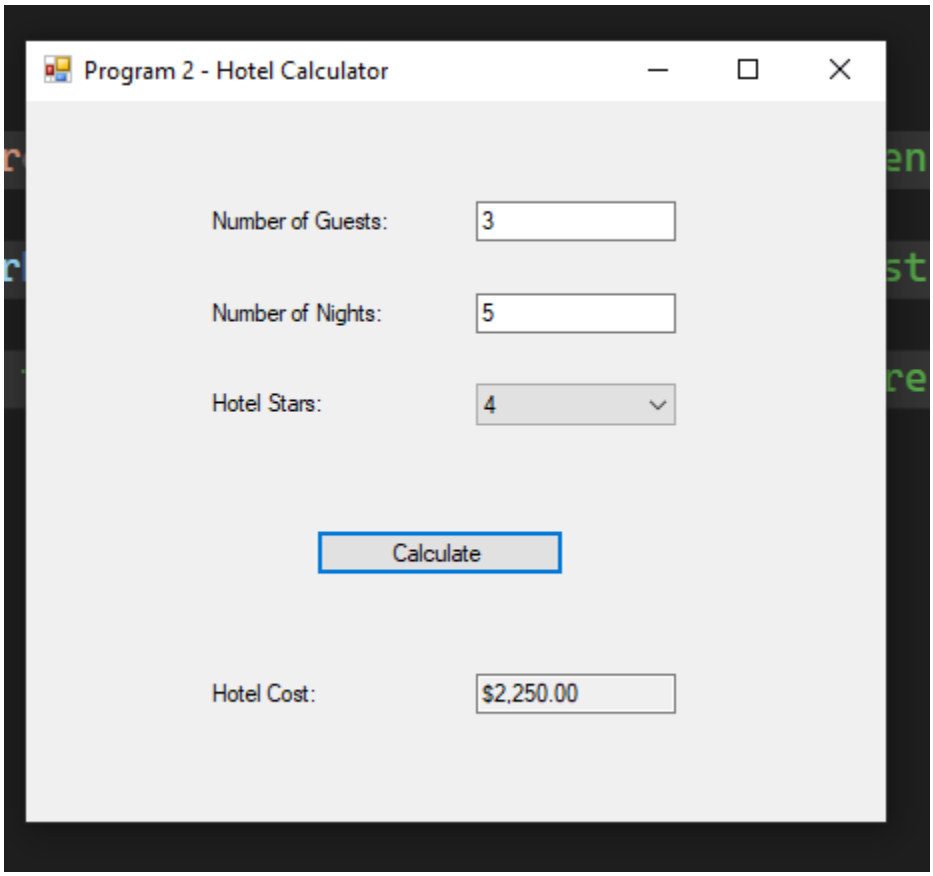


Program.cs:

```
//Lab 7
//K3215
//CIS199-01
//4/10/2022
//Console App to print stars.

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

namespace Lab7
{
   class Program
   {
      static void Main(string[] args)
      {
         int numStars = 0;
         bool value = false;
```

```
            Console.WriteLine("Enter number: ");
            while (value != true)
            {
                if (int.TryParse(Console.ReadLine(), out numStars) && numStars > 0) //user input and
validation
                {
                    ShowSquareOfStars(numStars); //display result from ShowSquareOfStars method
                    value = true;
                }
                else
                {
                    Console.WriteLine("Please enter valid positive integer: "); //error message if input
not valid
                }
            }
        }

        static void ShowSquareOfStars(int numStars)
        {
            for (int i = 0; i < numStars; i++) // loops for the number of rows printed
            {
                for (int j =0; j < numStars; j++) // loops for number of stars printed
                {
                    Console.Write("*");
                }
                Console.WriteLine("");
            }
        }
    }
}
```

File name: Prog2



Form1.cs:

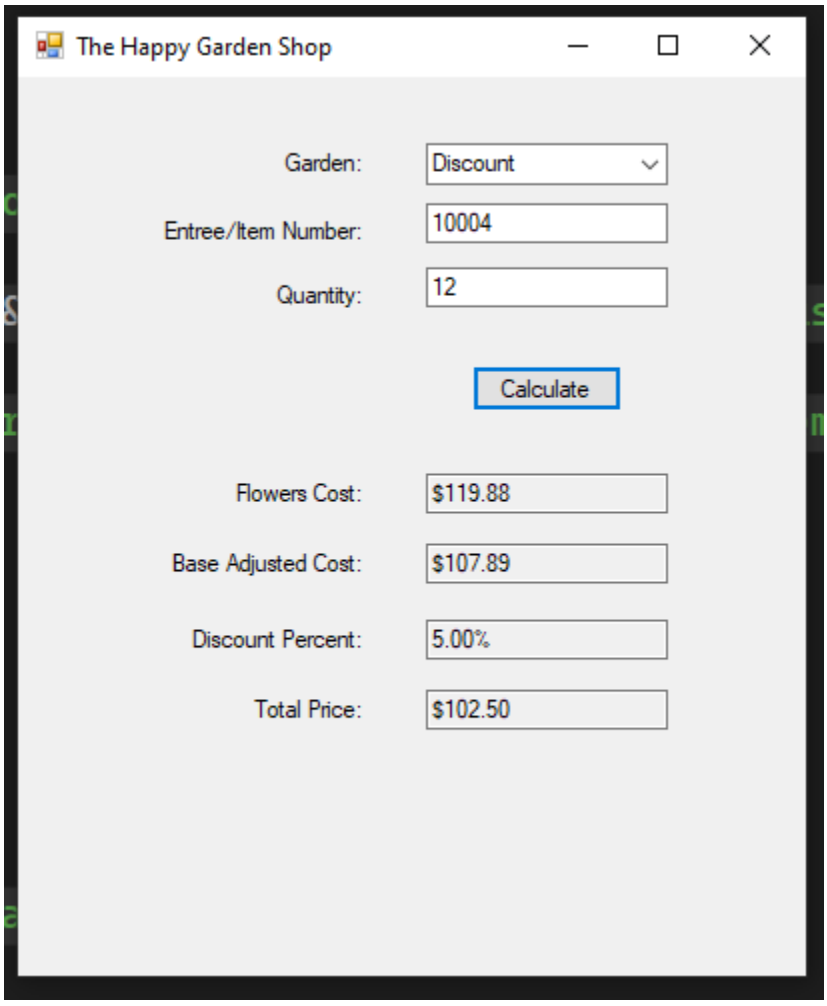
```
/*Program 2
  Grading ID: K3215
  Due Date: 3/10/22
  Course: CIS199-01
  Comment: Program to calculate hotel reservation cost depending on number of nights stayed, number of guest, and hotel star rating.
*/
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace Prog2
{
    public partial class programTwoForm : Form
    {
        public programTwoForm()
        {
            InitializeComponent();
        }

        // button to calculate the hotel cost
        private void calcButton_Click(object sender, EventArgs e)
        {
            //defining the variables
            int guest;
            int nights;
            int stars;
            double cost;
            int guestCost = 0;
            int nightsCost = 0;
            double starFactor = 0;

            //defining constant variables, price of per night stayed
            const int DAILY_RATE = 100;
            const int WEEKLY_RATE = 75;
            const int MONTHLY_RATE = 25;

            //input and conditions for number of guests
            if (int.TryParse(guestText.Text, out guest))
            {
                if (guest >=4 && guest <= 7) //for input value of 4-7 guests, the cost is 400
                    guestCost = 400;
                else if (guest == 3) //for input value of 3 guests, the cost is 250
                    guestCost = 250;
                else if (guest == 2) //for input value of 2 guests, the cost is 150
                    guestCost = 150;
                else if (guest == 1) //for input value of 1 guest, the cost is 100
                    guestCost = 100;
                else
                    MessageBox.Show("Invalid number of guests"); //message box for when guest
number is out of range, invalid input
            }
            else
                MessageBox.Show("Invalid number of guests, maximum guest number is 7");
//message box for when input is not an integer, invalid input
```

```
        //input and conditions for number of nights stayed
        if (int.TryParse(nightText.Text, out nights))
        {
            if (nights >= 31) //when staying over 31 nights, the cost per night is the monthly rate
                nightsCost = MONTHLY_RATE;
            else if (nights >= 7) //when staying over 7 nights but under 30 nights, the cost per
night is the weekly rate
                nightsCost = WEEKLY_RATE;
            else if (nights >= 1) //when staying over 1 night (inclusive) but under 7 nights (not
inclusive), the cost per night is the monthly rate
                nightsCost = DAILY_RATE;
            else
                MessageBox.Show("Invalid number of nights, please input a value over 1");
//message box for when number of nights stayed is out of range, invalid input
        }
        else
            MessageBox.Show("Invalid number of nights, please input a value over 1");
//message box for when input is not an integer, invalid input


        if (starsCombo.SelectedIndex > -1) //checking if a value is selected
        {
            if (int.TryParse(starsCombo.Text, out stars)) //value is an integer from the selection of
combobox
            {
                if (stars == 1) //if the hotel is 1 star, price multiply by 1
                    starFactor = 1;
                if (stars == 2) //if the hotel is 2 star, price multiply by 1.5
                    starFactor = 1.5;
                if (stars == 3) //if the hotel is 3 star, price multiply by 2.5
                    starFactor = 2.5;
                if (stars == 4) //if the hotel is 4 star, price multiply by 3
                    starFactor = 3;
                if (stars == 5) //if the hotel is 5 star, price multiply by 4
                    starFactor = 4;
            }
        }
        else
            MessageBox.Show("Please select hotel stars from dropdown"); //message box for
when no value of combobox dropdown is selected

        cost = (guestCost + (nights * nightsCost)) * starFactor; // calculating the total cost of
reservation
```

```
            costText.Text = cost.ToString("C"); //displaying the result cost on textbox with currency
format
        }
    }
}
```

File name: Prog3



Form1.cs:

```
// Program 3
// K3215
// CIS 199-01
```

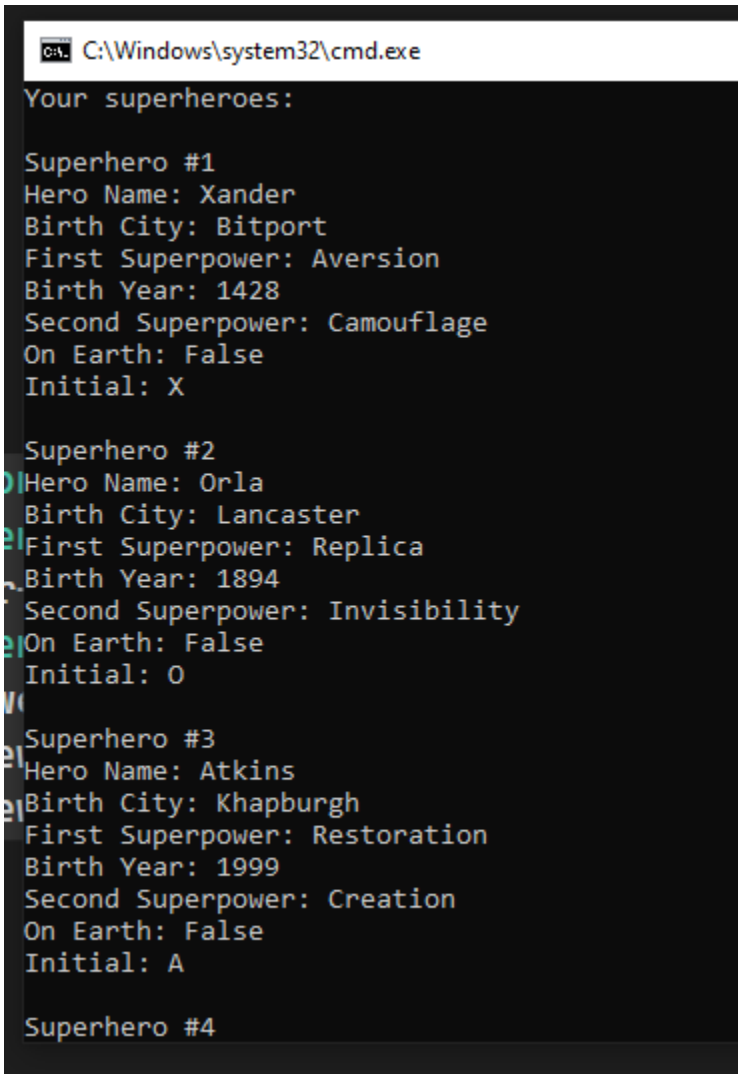
```
// 4/1/2022
// Form app to calculate costs for the garden shop.

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Prog3
{
    public partial class Program3 : Form
    {
        public Program3()
        {
            InitializeComponent();
        }

        //button to calculate result
        private void calcButton_Click(object sender, EventArgs e)
        {
            //arrays for given information
            string[] garden = { "Premium", "Standard", "Discount" };
            double[] baseRate = {1.1, 1.0, 0.9 };
            int[] flowerNumber = { 10001, 10002, 10003, 10004, 10005, 10006, 10007 };
            double[] costFlower = { 7.87, 9.51, 10.73, 9.99, 11.99, 5.00, 4.58 };
            int[] range = { 0, 6, 16, 21 };
            double[] discount = {0, 0.05, 0.10, 0.15 };

            //variables for user input
            int item;
            int qty = 0;

            //variables for array value, used during calculation
            double foundRate = 0;
            double foundCost = 0;
            double foundDiscount = 0;

            // bool variables set as control
            bool rateFound = false;
```

```
        bool costFound = false;
        bool discountFound = false;


        for (int h=0; h<garden.Length && !rateFound; h++) //loop for matching combobox items
with base rate
        {
            if (gardenCombo.SelectedIndex > -1) //checking if a value is selected
            {
                if (gardenCombo.Text == garden[h]) //if statement for when combobox input
matches a value in array
                {
                    rateFound = true;
                    foundRate = baseRate[h];
                    continue;
                }
            }
            else
            {
                MessageBox.Show("Please select garden type from dropdown"); //message box for
invalid input
                break;
            }
        }


        for (int i = 0; i < flowerNumber.Length && !costFound; i++) //loop to search through array
        {
            if (int.TryParse(itemText.Text, out item) && item >= 10001 && item <= 10007)
//checking for user input validity
            {
                if (item == flowerNumber[i]) //if statement for when user input is matched with array
item
                {
                    costFound = true;
                    foundCost = costFlower[i];
                    continue;
                }
            }
            else
            {
                MessageBox.Show("Input the correct item number.");//message box for invalid input
                break;
            }
```

```
        }

        for (int j = range.Length-1; j >= 0 && !discountFound; j--) // for loop to match ranges and
the designated discount
        {
            if (int.TryParse(quantityText.Text, out qty) && qty >= 0) //checking if input is valid
            {
                if (qty >= range[j]) //if statement for when input is larger or equal to the lower value
of range
                {
                    discountFound = true;
                    foundDiscount = discount[j];
                    continue;
                }
            }
            else
            {
                MessageBox.Show("Input a quantity number."); //message box for invalid input
                break;
            }
        }

        double flowerCostCalc = foundCost * qty ; //cost based on price and quantity of flower.
        double baseCostCalc = foundRate * flowerCostCalc; //cost based on the above times
the base rate according to garden type.
        double discountPercent = foundDiscount; //discount percent the customer is receiving
according to quantity.
        double totalPrice = baseCostCalc - (baseCostCalc * discountPercent); //total price based
on values above.

        //printing results to designated text box
        flowersCostText.Text = flowerCostCalc.ToString("C");
        baseCostText.Text = baseCostCalc.ToString("C");
        discountText.Text = discountPercent.ToString("P");
        totalText.Text = totalPrice.ToString("C");
    }
  }
}

/*while loop for last part, switched to for loop instead.
  //int j = range.Length - 1;
        while (j >= 0 && !discountFound)
        {
            if (int.TryParse(quantityText.Text, out qty) && qty >= 0) //checking if input is valid
```

```
    {
        if (qty >= range[j]) //when user input is larger or equal to array value
        {
            discountFound = true;
            foundDiscount = discount[j];
            continue;
        }
        else
        {
            j--;
        }
    }
    else
    {
        MessageBox.Show("Input a quantity number.");
    }
}*/
```

File name: Prog4



Program.cs:

```
// Program 4
// K3215
// CIS 199-01
// 4/18/2022
// Console application to create superheroes.

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

```
namespace Prog4
{
   class Program
   {
      static void Main(string[] args)
      {
         // 5 objects + hard coded test data, names and city are random, do not mind the
weirdness.
         Superhero hero1 = new Superhero("Xander", "Bitport", "Aversion", 1428, "Camouflage");
         Superhero hero2 = new Superhero("Orla", "Lancaster", "Replica", 1894, "Invisibility");
         Superhero hero3 = new Superhero("Atkins", "Khapburgh", "Restoration", -1417,
"Creation"); //since year is negative should pop up 1999
         Superhero hero4 = new Superhero("Scott", "Luimchester", "Camouflage", 2004,
"Echolocation");
         Superhero hero5 = new Superhero("Hull", "Anschester", "Poison", 1995, "Web");

         //store into array
         Superhero[] heroes = { hero1, hero2, hero3, hero4, hero5 };

         //For first set of data
         Console.WriteLine("Your superheroes: ");
         Console.WriteLine();
         printHeroes(heroes);

         //updating data
         hero1.SecondSuperpower = "Creation";
         hero1.OnPlanetEarth();
         hero2.OnPlanetEarth();
         hero3.FirstSuperpower = "Destruction";
         hero4.FirstSuperpower = "Manipulation";
         hero5.SecondSuperpower = "Teleport";

         //For updated data
         Console.WriteLine("Your updated superheroes: ");
         Console.WriteLine();
         printHeroes(heroes);

         //status all set on earth
         hero1.OnPlanetEarth();
         hero2.OnPlanetEarth();
         hero3.OnPlanetEarth();
         hero4.OnPlanetEarth();
         hero5.OnPlanetEarth();
```

```
            //For updated all on earth data
            Console.WriteLine("All superheroes on earth: ");
            Console.WriteLine();
            printHeroes(heroes);
        }


        //print out all heroes
        public static void printHeroes(Superhero[] heroes)
        {
            for (int i = 0; i < heroes.Length; i++)
            {
                Console.WriteLine("Superhero #" + (i+1));
                Console.WriteLine(heroes[i]);
            }
        }
    }
}
```

Superhero.cs:

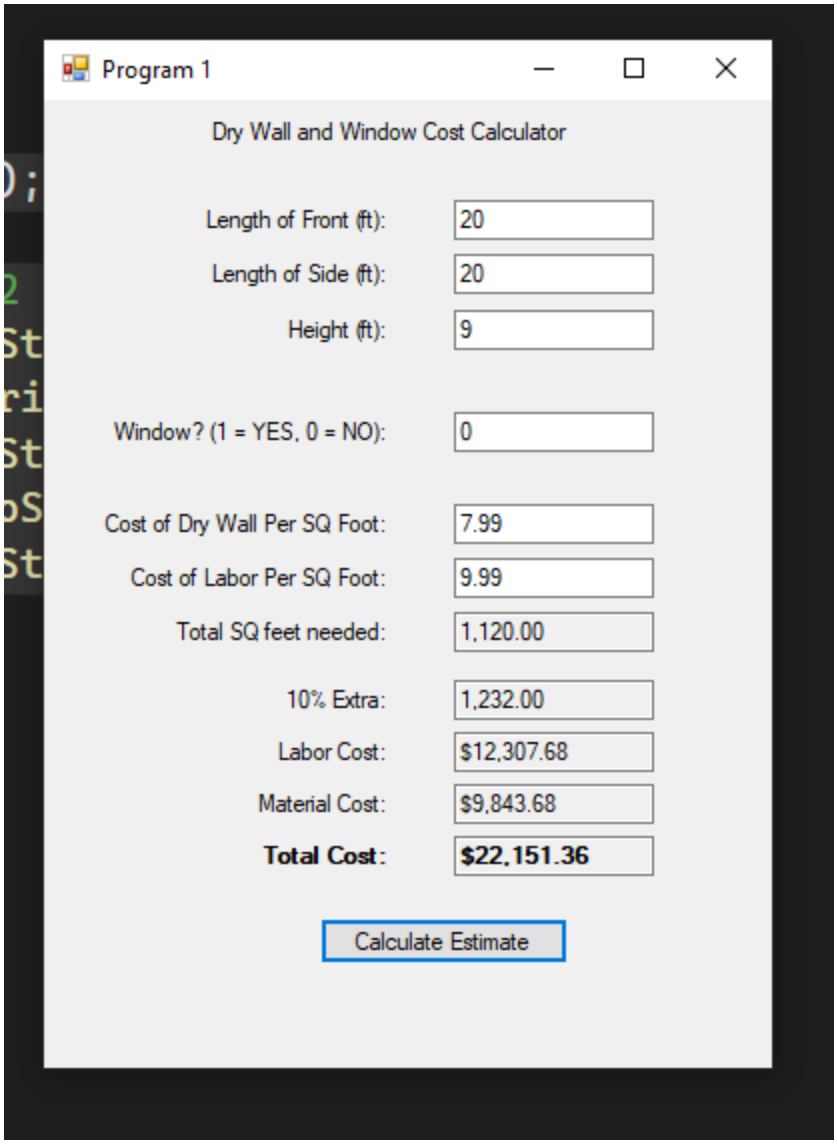
```
// Program 4
// K3215
// CIS 199-01
// 4/18/2022
// Console application to create superheroes.

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

namespace Prog4
{
    class Superhero
    {
        //defining objects
        private string _superheroName;
        private string _birthCity;
        private string _firstSuperpower;
        private int _birthYear;
```

```
        private string _secondSuperpower;
        private bool status;

        //5 parameter constructor, birth year 0+, status false since not on earth, valid values are
returned
        public Superhero(string superheroname, string birthcity, string firstsuperpower, int birthyear,
string secondsuperpower)
        {
            SuperheroName = superheroname;
            BirthCity = birthcity;
            FirstSuperpower = firstsuperpower;
            BirthYear = birthyear;
            SecondSuperpower = secondsuperpower;
            status = false;
        }

        //hero name property, data value only, no validation
        public string SuperheroName
        {
            get { return _superheroName; }
            set { _superheroName = value ; }
        }

        //birth city property, value only, no validation
        public string BirthCity
        {
            get { return _birthCity; }
            set { _birthCity = value ; }
        }

        //firrst superpower, value only, no validation
        public string FirstSuperpower
        {
            get { return _firstSuperpower; }
            set { _firstSuperpower = value ; }
        }

        //birth year property, value only, validation where it must be positive number else default is
1999
        public int BirthYear
        {
            get { return _birthYear; }
            set {
                    if (value >= 0)
```

```
            {
                _birthYear = value;
            }
            else
            {
                _birthYear = 1999;
            }
        }
}

//second superpower, value only, no validation.
public string SecondSuperpower
{
    get { return _secondSuperpower; }
    set { _secondSuperpower = value; }
}


//character property to return first character of superhero name property
public char Initial
{
    get { return SuperheroName[0]; }
}


//reflects status on whether character is on earth
public void OnPlanetEarth()
{
    status = true;
}

//reflects status on whether character is not on earth
public void OffPlanetEarth()
{
    status = false;
}

//returns the values true/false of status
public bool IsOnPlanetEarth()
{
    return status;
}

//ToString to format string details.
```

```
        public override string ToString()
        {
            string printData = "";

            printData += "Hero Name: " + SuperheroName + Environment.NewLine;
            printData += "Birth City: " + BirthCity + Environment.NewLine;
            printData += "First Superpower: " + FirstSuperpower+ Environment.NewLine;
            printData += "Birth Year: " + BirthYear + Environment.NewLine;
            printData += "Second Superpower: " + SecondSuperpower + Environment.NewLine;
            printData += "On Earth: " + status + Environment.NewLine;
            printData += "Initial: " + Initial + Environment.NewLine;

            return printData;
        }
    }

}
```

File name: Program 1



Form1.cs:

```
/* Program 1
   Grading ID: K3215
   Due Date: 2/15/22
   Course: CIS 199-01
   Comment: Form to calculate cost of building a shed.
 */
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Program_1
{
    public partial class programOne : Form
    {
        public programOne()
        {
            InitializeComponent();
        }

        // button to initiate the process of calculating the cost of building a shed.
        private void calculateButton_Click(object sender, EventArgs e)
        {
            // defining the variables.
            double frontLength;
            double sideLength;
            double height;
            int window;
            double dryWall;
            double costOfLabor;
            double totalFeet;
            double tenExtra;
            double laborCost;
            double matCost;
            double totalCost;

            // defining constant viariable.
            double TEN_EXTRA = 1.1;
            double WINDOW_FEE = 100;
            double WALL_SIDE = 2;

            // Read user input for each variable and convert to double or integer as needed.
            frontLength = double.Parse(frontLengthText.Text);
            sideLength = double.Parse(sideLengthText.Text);
            height = double.Parse(heightText.Text);
            window = int.Parse(windowText.Text);
            dryWall = double.Parse(wallCostText.Text);
            costOfLabor = double.Parse(laborPerFootText.Text);
```

```
        // calculation of total SQ feet needed, ten percent extra, labor cost, material cost, and the
final total cost.
        totalFeet = (frontLength * sideLength) + (frontLength * height * WALL_SIDE) +
(sideLength * height * WALL_SIDE);
        tenExtra = totalFeet * TEN_EXTRA;
        laborCost = costOfLabor * tenExtra;
        matCost = dryWall * tenExtra;
        totalCost = laborCost + matCost + (window * WINDOW_FEE);

        // display the result of calculation in text box with 2 decimal places.
        totalFeetText.Text = String.Format("{0}", totalFeet.ToString("N2"));
        tenExtraText.Text = String.Format("{0}", tenExtra.ToString("N2"));
        laborCostText.Text = String.Format("{0}", laborCost.ToString("C2"));
        materialCostText.Text = String.Format("{0}", matCost.ToString("C2"));
        totalCostText.Text = String.Format("{0}", totalCost.ToString("C2"));

    }
  }
}
```