



C# PROGRAMMING

(330)

REGIONAL 2023

PRODUCTION:

HorseRaceAppSetup

_____ (425 points)

Test Time: 90 minutes

Note To Graders:

- Run the solution code a few times to get a feel for how the app should operate.
- The results from the student product will NOT exactly match the results from the solution examples due to the randomization of the information.
- Check students code for evidence of hard coding the data for the creation of the horses.
- The only button that should be active at the start of the program is the ["Create Horses"].
- When adding a horse, if the user clicks on the ["Add Horse"] button without altering the default text, the only error that will occur is that the user needs to enter in a whole number. The program will recognize that "Name Entry" is an actual horse name.
- There could be a possibility of duplicated numbers randomly selected. This is acceptable. However, there should be NO duplications in the names.
- Students need to name the internal class Horse.cs.
- Students cannot add any additional buttons to the form design.
- Students are not allowed to change any object names.
- Student comments do not have to be complete sentences but there needs to be some relevance to the code it is commenting upon and they must use the comment code marker given in the rubric: NO exceptions.
- Students CANNOT alter the pre-created horse names.
- The ["Create Horses"] button should only work one time.

Input/Output

NOTE: the input and output for this program will be dynamic and will depend on which action the user wishes to take. However, the user input via clicking the [Create Horses] button will be the required action to access any of the other functionality of the program. The two panels below show the before and after from original creation of the list of horses.

Panel 1

Create Horses

Highest # MaxNumber

Lowest # MinNumber

Add Horse

Name Entry

Number Entry

Panel 2

Create Horses

Highest # MaxNumber

Lowest # MinNumber

Add Horse

Name Entry

Number Entry

1) Alex #63
2) Lucky #75
3) Dakota #72
4) Dakota #17
5) Daisy #0
6) Annie #0
7) Sunshine #37
8) Cricket #8
9) Cash #98
10) Tucker #22

*The user clicks on [Create Horses] button (**Panel 1**) and it causes the list box to display the horse names and numbers (**Panel 2**). In addition, buttons/text-boxes are disabled when their font and outline are faded. They will be enabled after the list is create. In **Panel 2** the [Create Horses] button is disabled; the user will only be able to use it 1 time.*

Panel 3

Create Horses

1) Alex #63
2) Lucky #75
3) Dakota #72
4) Dakota #17
5) Daisy #0
6) Annie #0
7) Sunshine #37
8) Cricket #8
9) Cash #98
10) Tucker #22

Highest # 98

Lowest # MinNumber

Panel 4

Create Horses

1) Alex #63
2) Lucky #75
3) Dakota #72
4) Dakota #17
5) Daisy #0
6) Annie #0
7) Sunshine #37
8) Cricket #8
9) Cash #98
10) Tucker #22

Highest # 98

Lowest # 0

***Panel 3 & 4** demonstrate the [Highest #] and [Lowest #] buttons finding the associated racing number from the list of horses. It should be noted, only the number is of interest and any duplicated numbers is of NO concern (they are randomly created)*

Panel 5

Highest # 98
Lowest # 0
Add Horse Mrs. Horse 1

9) Cash #98
10) Tucker #22
11) Mrs. Horse #1

Panel 5 demonstrates the user adding a new horse to the list. The list will automatically update the total count of horses. **NOTE:** to add the horse the [Add Horse] button must be pressed.

Panel 6 demonstrates the user adding a new horse to the list that might have a new highest # or lowest #. The functionality of the [Highest #] and [Lowest #] buttons will still be able to work once enabled.

Panel 6

Highest # 99
Lowest # 0
Add Horse New-High 99

8) Cricket #8
9) Cash #98
10) Tucker #22
11) Mrs. Horse #1
12) New-High #99

Panel 7

Highest # 99
Lowest # 0
Add Horse

8) Cricket #8
9) Cash #98
10) Tucker #22
11) Mrs. Horse #1
12) New-High #99

Please enter in a name and number
OK

Panel 7 demonstrates the user leaving the text boxes for name and number blank. This will cause a message box to appear requiring the user to enter in the appropriate information. If either one is blank, the message box must display.

Panel 8

Highest # 99
Lowest # 0
Add Horse Good Bad

8) Cricket #8
9) Cash #98
10) Tucker #22
11) Mrs. Horse #1
12) New-High #99

Please enter a whole number for the horse #
OK

Panel 8 demonstrates the user attempting to enter a non positive number into the number text box, causing the message box to appear. **NOTE:** this must appear if the user attempts to enter in letters, decimals, or negative values.

Solution and Project		
The project is present on the flash drive		10 points
Program Execution		
The program runs from the USB flash drive: the form loads with no initial errors		15 points
<i>If the program does not execute, then the remaining items in this section receive a score of zero.</i>		
The program displays only the ["Create Horses"] button as active; all other buttons and the text boxes are not active		10 points
When the ["Create Horses"] button is pressed the List Box displays 10 horses in the required format per the panel examples: i.e. 1) NAME #22		30 points
The ["Create Horses"] button is deactivated, and the ["Highest #"], ["Lowest #"] and ["Add Horse"] buttons and the "Name Entry" and "Number Entry" are ALL activated. NO partial credit.		10 points
When ["Highest #"] button is pressed the label will display the highest horse number from the list that was created with NO other effects on the program.		30 points
When ["Lowest #"] button is pressed the label will display the lowest horse number from the list that was created with NO other effects on the program.		20 points
When a proper name and number are entered into the text boxes, and the ["Add Horse"] button is pressed the horse is added as the 11 th horse to the list. When a second horse with proper information the horse is added as the 12 th horse in the list.		30 points
When either a horse name or number is not entered a message box appears instructing the user to "Please enter a name and number."		20 points
When a horse name is entered BUT the user enters in a negative number, decimals, or letters... a message box appears instructing the user to "Please enter a whole number for the horse #."		20 points
When ["Highest #"] button is pressed the label will display the highest horse number from the updated list that was created with NO other effects on the program. The user should be able to enter a new highest number and it will be displayed.		10 points
When ["Lowest #"] button is pressed the label will display the lowest horse number from the updated list that was created with NO other effects on the program. The user should be able to enter a new lowest number and it will be displayed.		10 points
Output matches required format.		20 points
Subtotal		/235 Points

Source Code Review (NOTE The code must have worked to get credit)		
Form1.cs a comment containing the contestant number is present		10 points
Horse.cs class: contains a constructor that accepts a String and integer parameter and assigns it to acceptable instance variables. Mark this comment as "SC1" at the beginning.		10 points
Horse.cs class: contains two get methods. getName () returns a string & getNumber () returns an integer. These must be accessible to the main form. Mark this comment as "SC2" at the beginning.		10 points
Horse.cs class: contains two set methods. setName (String) updates the object name & setNumber (integer) updates the object number. These must be accessible to the main form. Mark this comment as "SC3" at the beginning.		10 points
Horse.cs class: contains a method that returns a string in the following format "NAME # HORSENUMBER". i.e. "Daisy #23" This must be accessible to the main form. Student has freedom to name this method. Mark this comment as "SC4" at the beginning.		10 points
Form1.cs class: contains a dynamic list of Horse objects. Mark this comment as "SC5" at the beginning.		10 points
Form1.cs class: when button btnCreate is clicked it will call the setHorse () method to create 10 random horse object combinations: the name from the provided list must be randomly selected, and a random number from 0 to 99 (inclusive) and adds them to the list. Mark this comment as "SC6" at the beginning.		30 points
Form1.cs class: when button btnCreate is clicked it will add the horses to the ListBox listHorses1. Mark this comment as "SC7" at the beginning.		10 points
Form1.cs class: when button btnCreate is clicked it will enable all other buttons and text boxes, and disable the button btnCreate . Mark this comment as "SC8" at the beginning.		10 points
Form1.cs class: when button btnMax is clicked it will search the list of horses and find the horse with the highest number value and display it in the appropriate label. Mark this comment as "SC9" at the beginning.		30 points
Form1.cs class: when button btnMin is clicked it will search the list of horses and find the horse with the lowest number value and display it in the appropriate label. Mark this comment as "SC10" at the beginning.		10 points
Form1.cs class: when button btnAddHorse is clicked it will contain code (hint: bool TryParse(string s, out int result)) to screen inputted numbers into the textbox txtNumber to ensure they are only positive whole numbers. No non numbers, negative, or decimal values. Mark this comment as "SC11" at the beginning.		20 points
Form1.cs class: when button btnAddHorse is clicked it will contain code (hint: bool IsNullOrEmpty(string value)) to screen inputted names into the textbox txtName to ensure that a value is entered (it can be any alpha-numeric or symbol). Mark this comment as "SC12" at the beginning.		20 points
Subtotal		/190 Points
Total Points		/425 Points

```

...Regional\HorseRaceAppSetup\HorseRaceAppSetup\Form1.cs 1
1 using System;
2 using System.Collections.Generic;
3 using System.Collections;
4 using System.ComponentModel;
5 using System.Data;
6 using System.Drawing;
7 using System.Linq;
8 using System.Text;
9 using System.Threading.Tasks;
10 using System.Windows.Forms;
11
12 namespace HorseRaceAppSetup
13 {
14     public partial class Form1 : Form
15     {
16         //Student Point
17         List<Horse> horses = new List<Horse>();
18         //-----names will be given as comments but have to get made into
19         string array
20         string [] names =
21         {"Bella", "Sugar", "Alex", "Alexia", "Lady", "Tucker", "Fancy", "Cash",
22         "Dakota", "Daisy", "Spirit", "Cisco", "Annie", "Buddy", "Chance", "Dall
23         as", "Star", "Scout", "Lucky", "LadyBug", "Stinky", "Cricket", "Magic",
24         "Red", "Bruno", "Sunshine", "Storm", "Rose", "Storm", "Cloud" };
25
26         //Student Point
27         private void setHorse()
28         {
29             var rand = new Random();
30             for (int i = 0; i < 10; i++)
31             {
32                 Horse h = new Horse(names[rand.Next(names.Length)],
33                 rand.Next(100));
34
35                 horses.Add(h);
36             }
37         }
38         //Given
39         public Form1()
40         {
41             InitializeComponent();
42         }
43         //Student Point Partial
44         private void Form1_Load(object sender, EventArgs e)
45         {
46             //Student Point
47             btnMax_.Enabled = false;
48         }
49     }
50 }

```

```

...Regional\HorseRaceAppSetup\HorseRaceAppSetup\Form1.cs 2
43         btnMin.Enabled = false;
44         btnAddHorse.Enabled = false;
45         txtName.Enabled = false;
46         txtNumber.Enabled = false;
47     }
48
49     //Student Point x2
50     private void btnCreate_Click(object sender, EventArgs e)
51     {
52         int i = 1;
53         setHorse();
54         foreach (Horse h in horses)
55         {
56             listHorses1.Items.Add(i+" " + h.getHorseInfo());
57             i++;
58         }
59
60         Console.WriteLine("Test Run"); //REMOVE
61         btnMax_.Enabled = true;
62         btnMin.Enabled = true;
63         btnAddHorse.Enabled = true;
64         btnCreate.Enabled = false;
65         txtName.Enabled = true;
66         txtNumber.Enabled = true;
67     }
68
69     //Student Point
70     private void btnMax__Click(object sender, EventArgs e)
71     {
72
73         int ind = 0, num =0; //horses.ElementAt(1).getNumber();
74
75         for (int i = 1; i < horses.Count; i++)
76         {
77             if (horses.ElementAt(i).getNumber() > horses.ElementAt
78                 (ind).getNumber())
79             {
80                 ind = i;
81                 num = horses.ElementAt(ind).getNumber();
82                 //REMOVE
83                 Console.WriteLine("ind" + ind + " num " + num + "
84                     Horse " + horses.ElementAt(ind).getNumber()+"\n");
85             }
86         }
87         lblMax.Text = horses.ElementAt(ind).getNumber().ToString();
88     }
89     //Student Point

```



```

...Regional\HorseRaceAppSetup\HorseRaceAppSetup\Form1.cs 3
90     private void btnMin_Click(object sender, EventArgs e)
91     {
92         int ind = 0, num = 0; //horses.ElementAt(1).getNumber();
93
94         for (int i = 1; i < horses.Count; i++)
95         {
96             if (horses.ElementAt(i).getNumber() < horses.ElementAt
97                 (ind).getNumber())
98             {
99                 ind = i;
100                num = horses.ElementAt(ind).getNumber();
101                //REMOVE
102                Console.WriteLine("ind" + ind + " num " + num + "
103                Horse " + horses.ElementAt(ind).getNumber() + "\n");
104            }
105        }
106        lblMin.Text = horses.ElementAt(ind).getNumber().ToString();
107    }
108    //Student Point x3
109    private void btnAddHorse_Click(object sender, EventArgs e)
110    {
111        int temp = 0;
112        bool result = int.TryParse(txtNumber.Text, out temp);
113        if (String.IsNullOrEmpty(txtName.Text) || String.IsNullOrEmpty
114            (txtNumber.Text) )
115        {
116            MessageBox.Show("Please enter in a name and number");
117            txtName.Clear();
118            txtNumber.Clear();
119            txtName.Focus();
120        }
121        else if (!result || temp < 0)
122        {
123            MessageBox.Show("Please enter a whole number for the horse
124                #");
125            txtNumber.Clear();
126            txtNumber.Focus();
127        }
128        else
129        {
130            listHorses1.Items.Clear();
131            Horse userHorse = new Horse(txtName.Text, int.Parse
132                (txtNumber.Text));
133            horses.Add(userHorse);
134            int i = 1;

```

```
...Regional\HorseRaceAppSetup\HorseRaceAppSetup\Form1.cs 4
134         foreach (Horse h in horses)
135         {
136             listHorses1.Items.Add(i + ") " + h.getHorseInfo());
137             i++;
138         }
139
140         txtName.Clear();
141         txtNumber.Clear();
142     }
143
144 }
145
146 }
147
```

```
...Regional\HorseRaceAppSetup\HorseRaceAppSetup\Horse.cs 1
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace HorseRaceAppSetup
8 {
9     internal class Horse
10    {
11        //Student Point
12        string name;
13        int number;
14
15        //Student Point
16        public Horse()
17        {
18            name = "default";
19            number = 0;
20        }
21
22        //Student Point
23        public Horse(string name, int number)
24        {
25            this.name = name;
26            this.number = number;
27        }
28        //Student Point
29        public string getName()
30        {
31            return name;
32        }
33        //Student Point
34        public int getNumber()
35        {
36            return number;
37        }
38
39        //Student Point
40        public string getHorseInfo()
41        {
42            return name + " #" + number;
43        }
44
45        //Student Point
46        public void setName(string s)
47        {
48            name = s;
49        }
50    }
```

```
...Regional\HorseRaceAppSetup\HorseRaceAppSetup\Horse.cs 2
50      //Student Point
51      public void setNumber(int i)
52      {
53          number= i;
54      }
55  }
56 }
57
```