

Member ID: \_\_\_\_\_

Time: \_\_\_\_\_

Rank: \_\_\_\_\_



# C# PROGRAMMING

## (330)

## REGIONAL 2023

**PRODUCTION:**

HorseRaceAppSetup

\_\_\_\_\_ (425 points)

**Test Time: 90 minutes**

**GENERAL GUIDELINES:**

*Failure to adhere to any of the following rules will result in disqualification:*

1. Member must hand in this test booklet and all printouts if any. Failure to do so will result in disqualification.
2. No equipment, supplies, or materials other than those specified for this event are allowed in the testing area. No previous BPA tests and/or sample tests (handwritten, photocopied, or keyed) are allowed in the testing area.
3. Electronic devices will be monitored according to ACT standards.

You will have ninety (90) minutes to complete your work.

Your name and/or school name should *not* appear on work you submit for grading.

1. Create a folder on the flash drive provided using your contestant number as the name of the folder.
2. Copy your entire solution/project into this folder. The project folder for you has already been provided: HorseRaceAppSetup
3. Submit your entire solution/project so that the graders may open your project to review the source code.
4. Ensure that the files required to run your program are present and will execute on the flash drive provided.
5. You will need to use Visual Studio to complete this exam.

\*Note that the flash drive letter may *not* be the same when the program is graded as it was when you created the program.

\*It is recommended that you use relative paths rather than absolute paths to ensure that the program will run regardless of the flash drive letter. It is **HIGHLY** recommended that you place all of the files into one folder.

The graders will *not* compile or alter your source code to correct for this.  
Submissions that do *not* contain source code will *not* be graded.

**Assumptions to make when taking this assessment:**

- The form design is already created and does not need to be adjusted.
- Users can attempt to enter in erroneous information during the input prompt for creating new horses.
- The program will only exit when the VS stop button is manually clicked.
- To create an internal class choose “Project” from the menu, and then select “Add Class”
- To quickly create a function for the button, double click on the button object in the design mode and it will be created within the partial class Form1.
- The names of the horses have been provided; the numbers will be randomly created from a range of 0 (zero) to 99.
- All required packages needed to complete have been provided for you

**Development Standards:**

- Your Code must use a consistent variable naming convention.
- All subroutines (if any), functions (if any), and methods (if any) must be documented with comments explaining the purpose of the method, the input parameters (if any), and the output (if any). Readability is a goal of good code.

## **HorseRaceAppSetup Overview**

In this challenge you will be programming a few of the base logic algorithms to be used in creating a horse racing game. The program at this point is limited to only data manipulation. When completed, the program will create 10 randomly generated horses based upon a list of names and then randomly creating a race number for them. For example, the program could create a horse called “Rose #23”, etc. The name “Rose” comes from a list, and the “#23” is randomly generated. The ten horses will be created displayed in the provided list box once “Create Horses” button is pressed. In addition, the program will find the horse from the list with the highest racing number, and also the lowest racing numbers. The results will be displayed in the corresponding label once the buttons are pressed.

Finally, the user will be able to add a horse to the list to increase the total count of horses beyond the 10 randomly generated horses. The list should be updated to reflect this addition. The data entered into the text boxes will have requirements on what is accepted as a valid entry: for example, the user cannot enter in letters for the horse number or numbers that are beyond a range of value, or if the text box is left blank. If a user enters a number for a name that is acceptable, and it is also acceptable if the user enters in duplicate names or numbers.

When the program first launches, the user should only be able to access the “Create Horses” button and the others will be disabled. After the “Create Horses” has been pressed it will become disabled and the other buttons will be enabled.

Horses that are created for the list and through the user input, will be required to be created with an internal object class called “Horse”. This internal class can easily be created by choosing “Project” from the menu, and then selecting “Add Class”. After that, name the class “Horse”.

## 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.

## Requirements

1. You must use the provided project called “RaceHorseAppSetup”. This project will contain a “Form1.cs[design]” that has been already created in the organization and naming conventions.
2. The file “Form1.cs” has already been provided. This file contains all needed packages and the method headers have already been provided. You will program within this file.
3. No additional buttons or methods can be added to the form design.
4. Your contestant number must appear as a comment at the top of the main source code file.
5. If incorrect data is entered, then the program should display an appropriate/similar message as shown in Panels 7 & 8 above.
6. You will create an internal class file called “Horse.cs”. You will program this entire object class file. No method headers have been provided however the names of the methods have been provided in the rubric (see next page). This file will automatically be stored in the project folder.
7. No random numbers for the horse number can exceed 99 (  $0 \leq \text{HorseNumber} < 100$  ).
8. NOTE: the generic text already placed in the text boxes will have to be removed by the user to enter in a horse to add. NO coding is required to change this text other than enabling the text box.
9. The program will display the similar outputs as shown in the examples above. NOTE: the horse name and number will NOT match the output in the sample due to the randomization of the information.
10. If a scorable item on the rubric says “Comments are required” you will only be able to earn points if that item has a comment with the comment code provided in the rubric. For example the rubric says “Mark this comment as “SC1” at the beginning” your comment should begin as such //SC1... ***Failure to use the comment code will result in loss of points.***
11. Comments do not need to be a complete sentence, and only needs to bring the graders attention to the main idea of what that method or segment of code performs. You do not need to list a comment for line by line explanation; however, your comment MUST have relevance to the code it is describing. NO generic comments.
12. WARNING: all buttons have been named in advance and should not be change. They will be prefixed with “btn”+Name. For example, ***btnMax***. In addition, the buttons have an event associated with them, and those methods have already been created in advance. The naming convention for these button events which you will program are named. For example the method for the button ***btnMax*** has the method ***private void btnMax\_Click(object sender, EventArgs e) {...}*** The ***\_Click(object sender, EventArgs e)*** has been added for you. You will program these method based upon the requirements in the rubric (see next page). All buttons in this program will follow these guidelines, and you will program ALL of the buttons. A list of all buttons and methods is provided below.
  - a. Button Name: ***btnCreate*** Method Name: ***btnCreate\_Click*** Text: “Create Horse”
  - b. Button Name: ***btnMax*** Method Name: ***btnMax\_Click*** Text: “Highest #”
  - c. Button Name: ***btnMin*** Method Name: ***btnMin\_Click*** Text: “Lowest #”
  - d. Button Name: ***btnAddHorse*** Method Name: ***btnAddHorse\_Click*** Text: “Add Horse”

<b>Solution and Project</b>		
The project is present on the flash drive		10 points
<b>Program Execution</b>		
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 <sup>th</sup> horse to the list. When a second horse with proper information the horse is added as the 12 <sup>th</sup> 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
<b>Subtotal</b>		<b>/235 Points</b>



<b>Source Code Review</b> ( <i>NOTE The code must have worked to get credit all items below must be commented</i> )		
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. <b>getName ()</b> returns a string & <b>getNumber ()</b> 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. <b>setName (String)</b> updates the object name & <b>setNumber (integer)</b> 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 <b>btnCreate</b> is clicked it will call the <b>setHorse ()</b> 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 <b>btnCreate</b> 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 <b>btnCreate</b> is clicked it will enable all other buttons and text boxes, and disable the button <b>btnCreate</b> . Mark this comment as "SC8" at the beginning.		10 points
Form1.cs class: when button <b>btnMax</b> 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 <b>btnMin</b> 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 <b>btnAddHorse</b> is clicked it will contain code (hint: <b>bool TryParse(string s, out int result)</b> ) 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 <b>btnAddHorse</b> is clicked it will contain code (hint: <b>bool IsNullOrEmpty(string value)</b> ) 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
<b>Subtotal</b>		<b>/190 Points</b>
<b>Total Points</b>		<b>/425 Points</b>