Assign 06

Due date and time: 04/25/2023 11:59 pm **Total points: 20**

Description: In this assignment, you will create a multiform desktop application that let the user maintain a list of student scores. You will use a text file to store the student data.

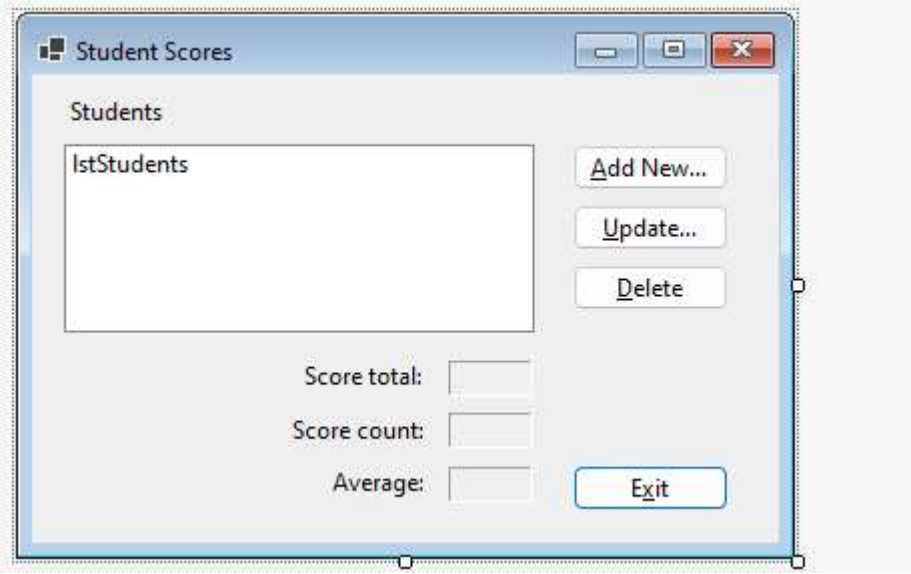
Specifications

- When this application starts, it should read student data from a text file named StudentScores.txt in the C:\C#\Files directory and display this data in the Students list box. If this directory or file doesn't exist, it should be created.
- When students are added, updated, or deleted, this application should write the student data to the StudentScores.txt file, overwriting any existing data.
- This application should use a class named StudentDB to read data from and write data to the StudentScores.txt file.
- The students should be displayed in alphabetical order in the list box.
- This application should use a class named Student to store information about each student.
- The Student class should override the ToString() method of the Object class to provide a string representation of each Student object that includes the student name and scores as shown above.
- The Update Student Scores form should create a clone of the current student object and then apply changes to the clone. That way, the changes will be saved to the current student only if the user clicks the OK button. To create a clone, the Student class will need to implement the ICloneable interface, and the Clone() method will need to implement a deep copy.
- This application should make sure that the user enters a name for a new student. However, a new student can be added without any scores.
- This application should check all scores entered by the user to make sure that each score is a valid integer from 0 to 100.
- Add data validation.
  - Required field, numeric, range 0-100.

- Create a separate Validator.cs static class to write data validation code.

- Add exception handling.

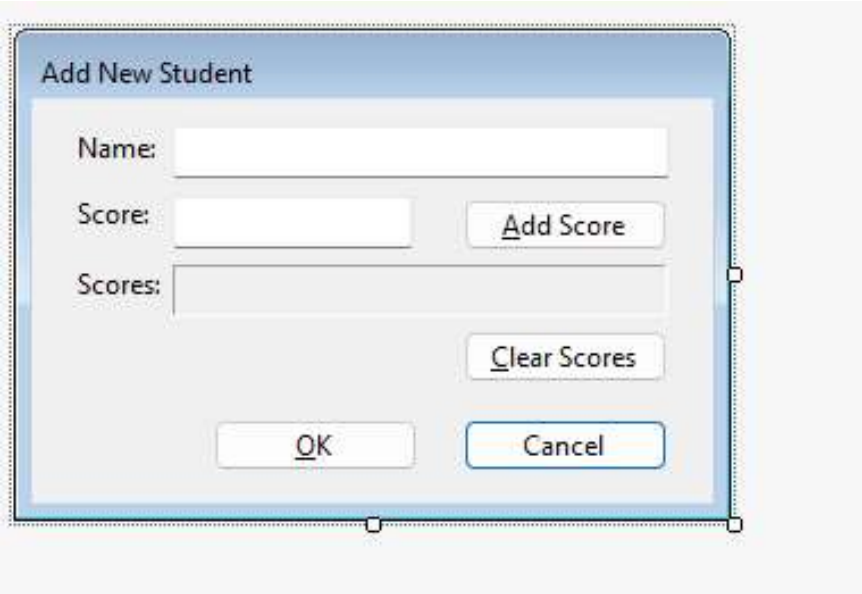
The Student Scores form



Operation:

- To display the total, count, and average for a student, the user selects the student from the list box. If the list box is empty, the total, count, and average labels should be cleared.
- To add a new student, the user clicks the Add New button to display the Add New Student dialog box.
- To update an existing student's scores, the user selects the student in the list box and clicks the Update button to display the Update Student Scores dialog box.
- To delete a student, the user selects the student in the list box and clicks the Delete button.

Add New student form.

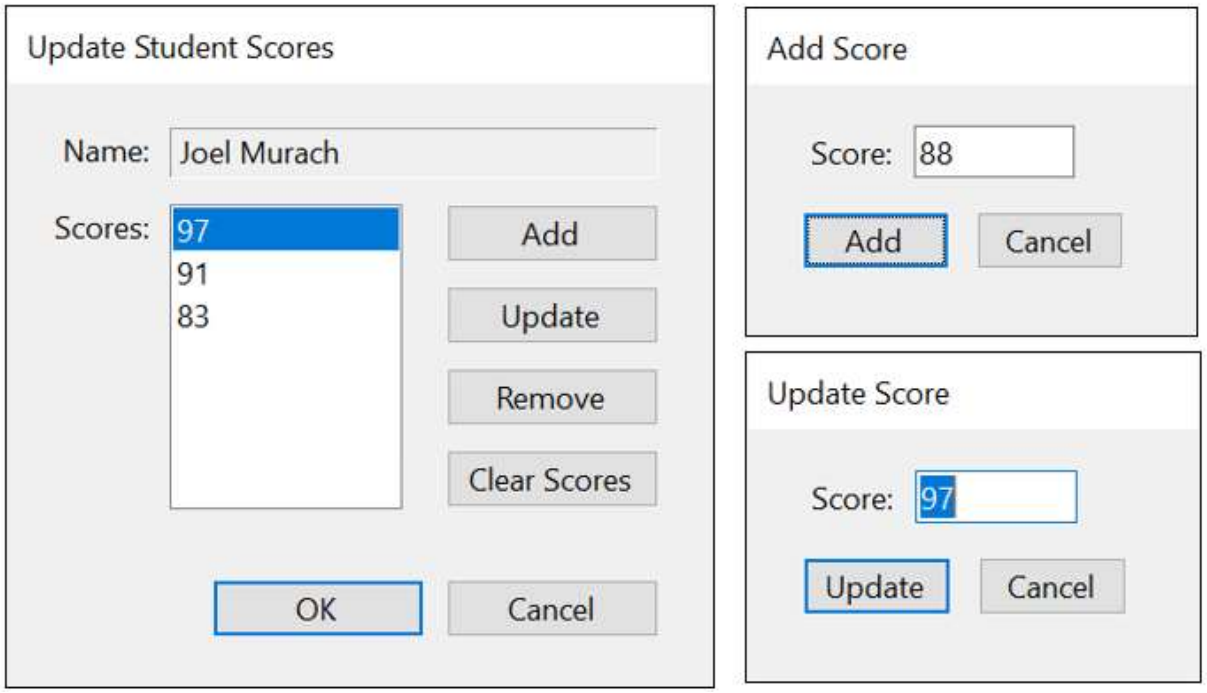


Operation

- To add a new student, the user enters a student name and, optionally, one or more scores and clicks the OK button.
- To add a score, the user enters a score and clicks the Add Score button. The score is added to the list of scores in the Scores label.
- To remove all scores from the Scores label, the user clicks the Clear Scores button.
- To cancel the add operation, the user clicks the Cancel button.

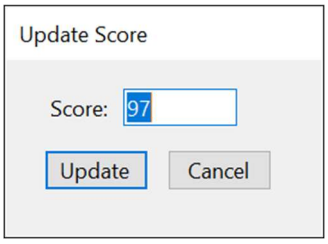
The Update Student Scores and Add/Update Score forms.

After clicking Update on Student Scores form, Click Add on Update Student Scores form, then add a score on Add Score.



Operation

- To add a score, the user clicks the Add button and enters the score in the Add Score dialog box that's displayed.

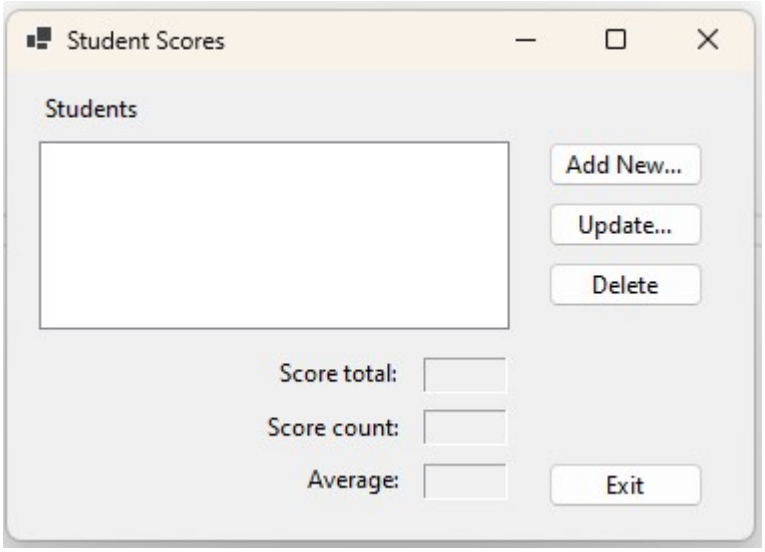


- To update a score, the user selects the score, clicks the Update button, and changes the score in the Update Score dialog box that's displayed.
- To remove a score from the Scores list box, the user selects the score and clicks the Remove button.
- To remove all scores from the Scores list box, the user clicks the Clear Scores button.
- To accept all changes, the user clicks the OK button.
- To cancel the update operation, the user clicks the Cancel button.

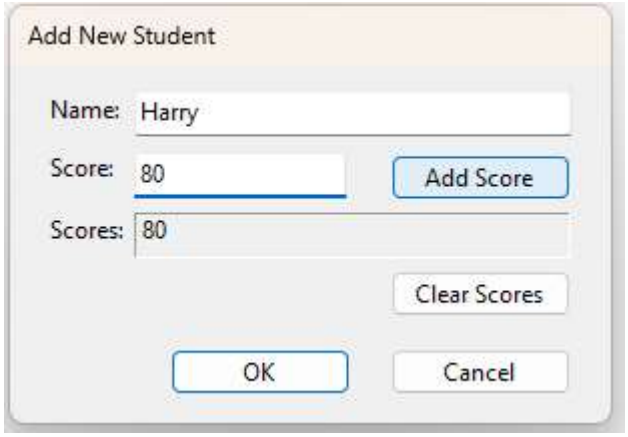
Hint

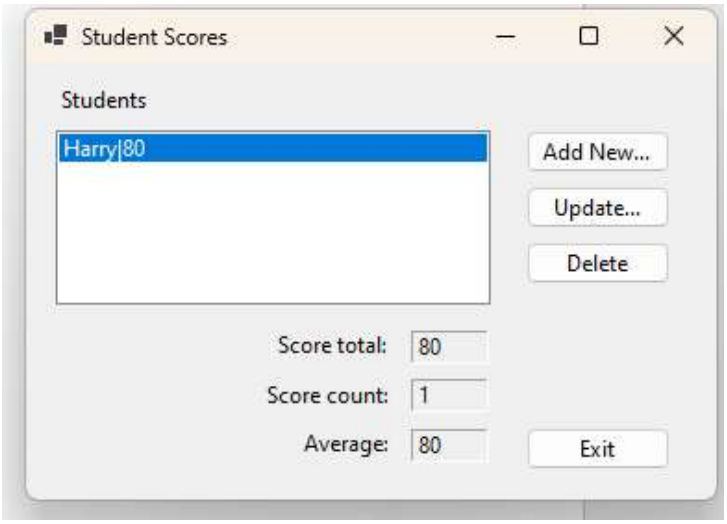
- To get the labels that display the score total, score count, and average on the Student Scores form, the scores on the Add New Student form, and the name on the Update Student Scores form to look like the ones shown above, you'll need to set the AutoSize property of the labels to False and the BorderStyle property to Fixed3D.

When you run this application – happy scenario.

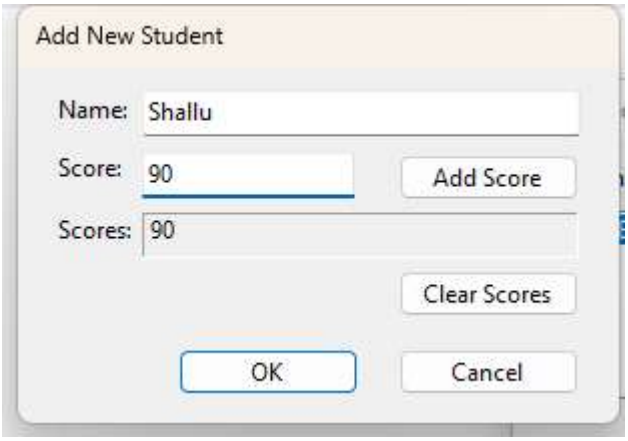


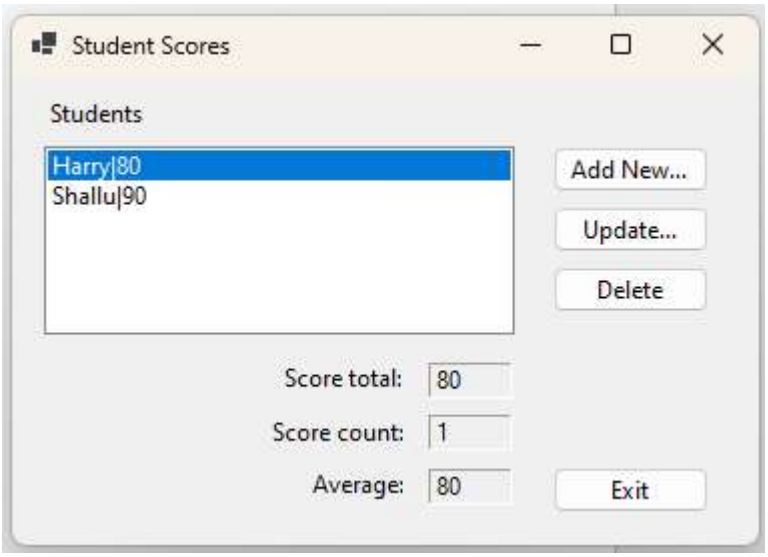
Click Add New.



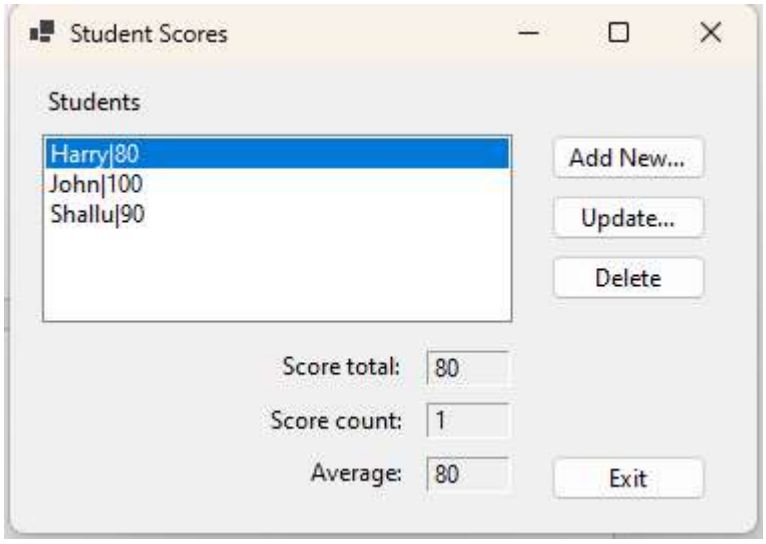


Add one more student.

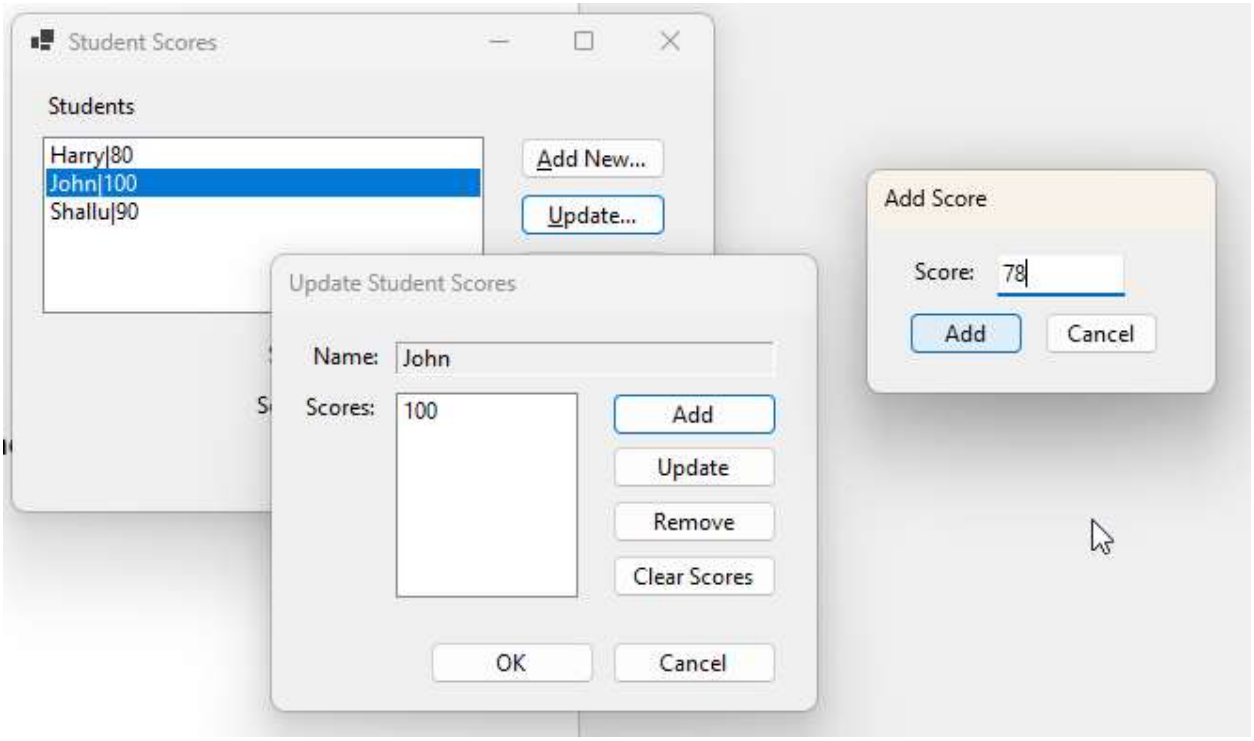




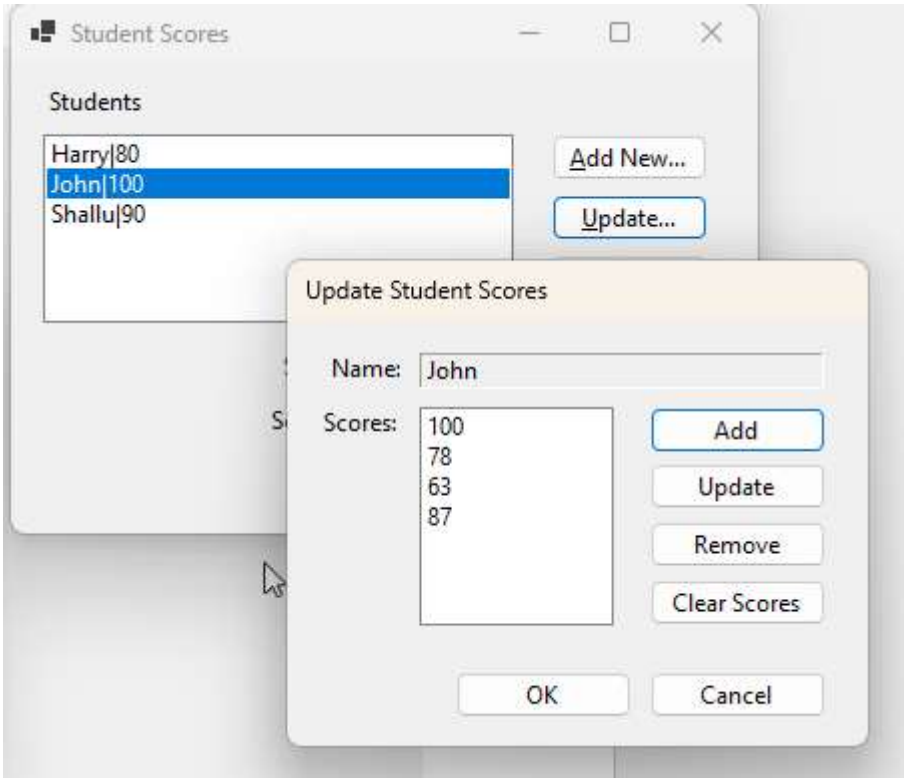
After adding John – pay attention to the sorting of names.

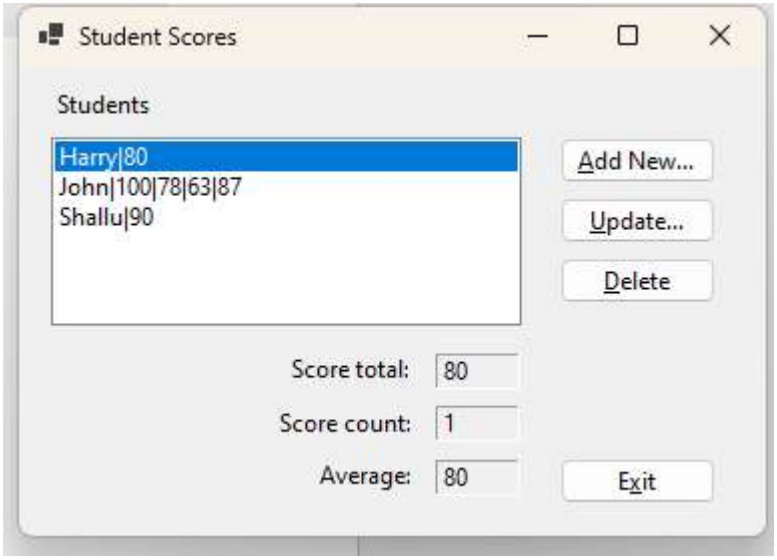


Let's update and add one more score to John

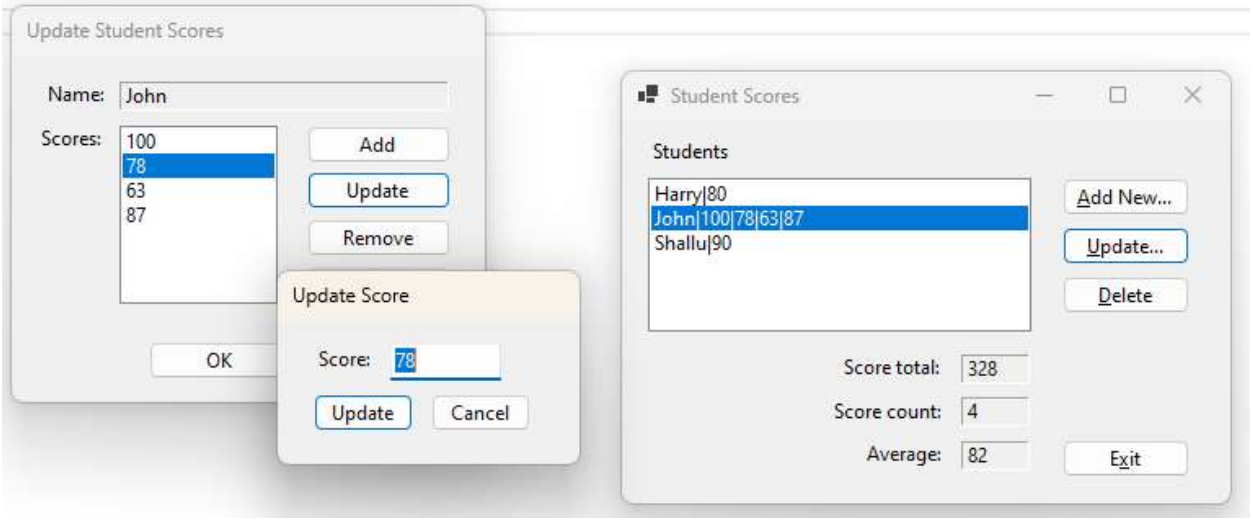


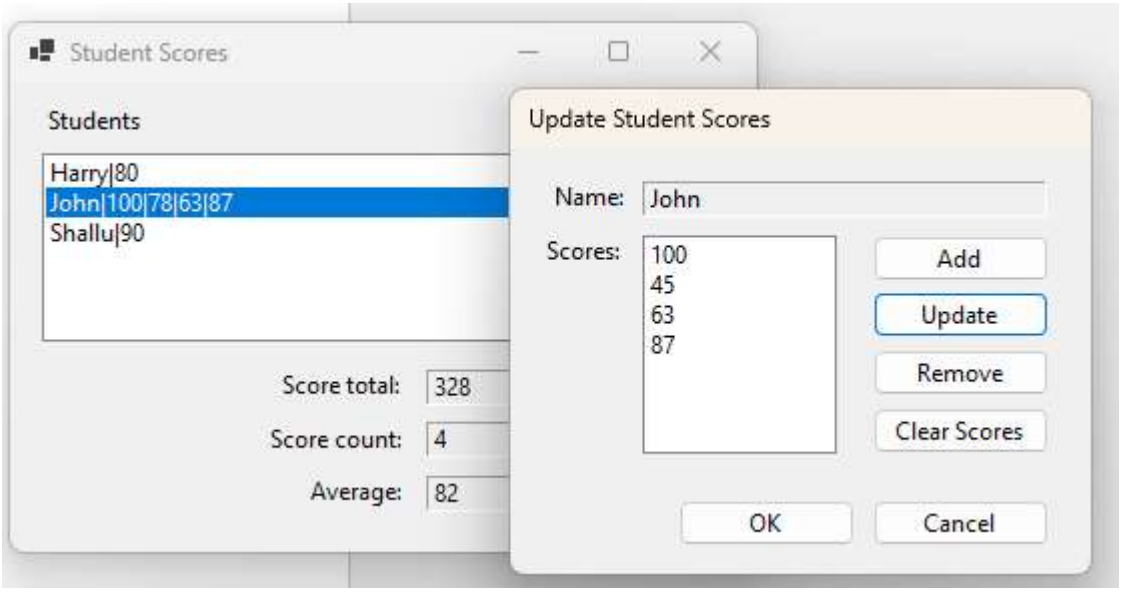
After adding a few more scores.

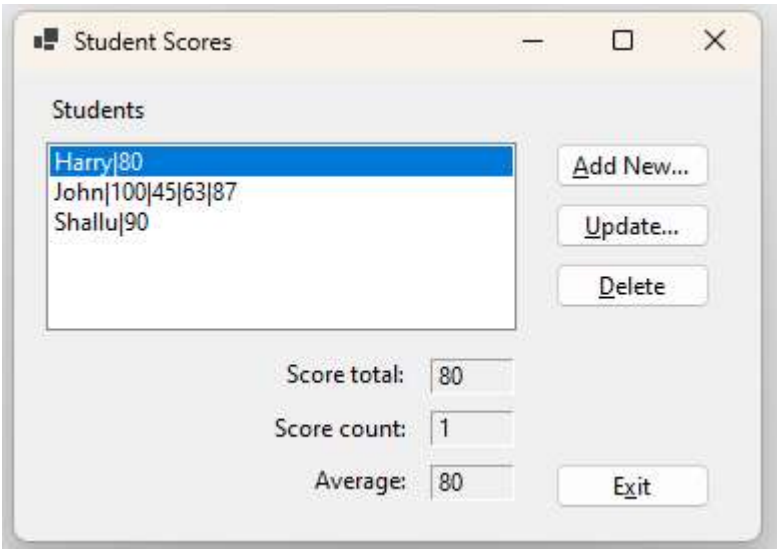




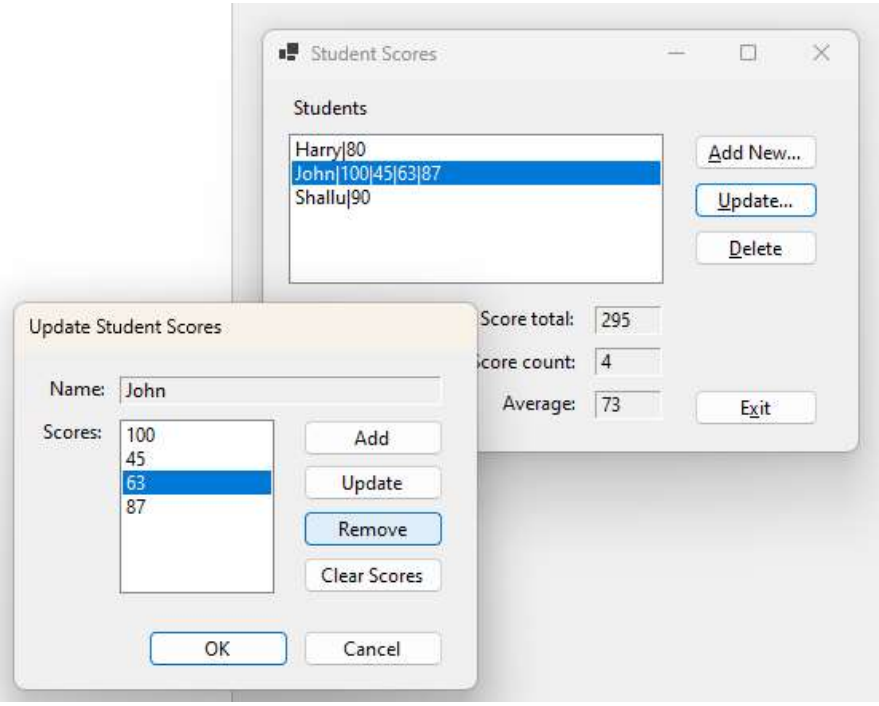
Let's update score 78 to 45 for Student John.

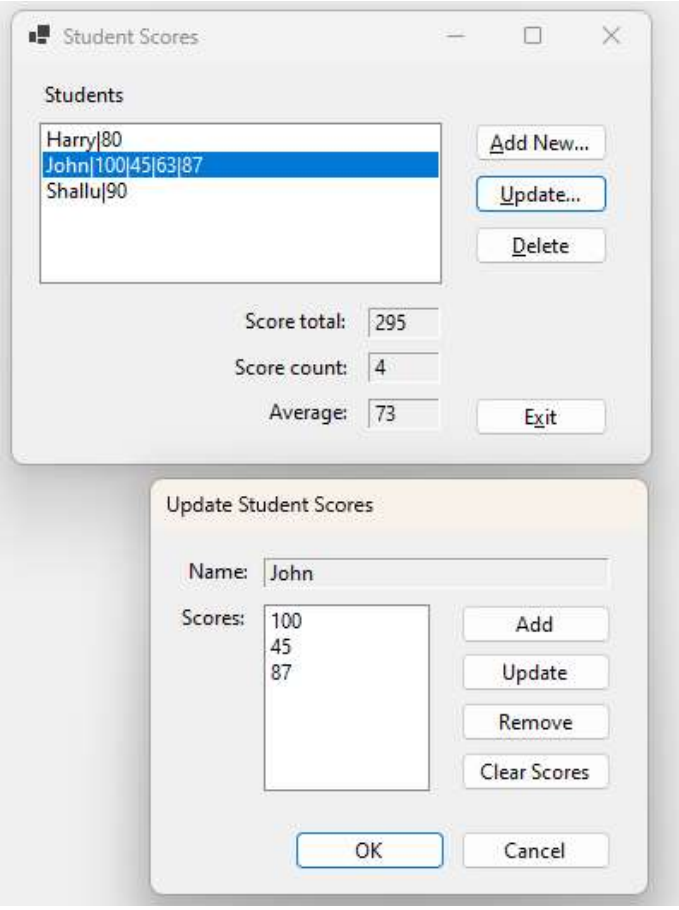

Update Student Scores
Name: John
Scores: 100 78 63 87
Add
Update
Remove
OK
Update Score
Score: 78
Update
Cancel
Student Scores
Students
Harry|80
John|100|78|63|87
Shallu|90
Add New...
Update...
Delete
Score total: 328
Score count: 4
Average: 82
Exit


Student Scores
Students
Harry|80
John|100|78|63|87
Shallu|90
Score total: 328
Score count: 4
Average: 82
Update Student Scores
Name: John
Scores: 100 45 63 87
Add
Update
Remove
Clear Scores
OK
Cancel

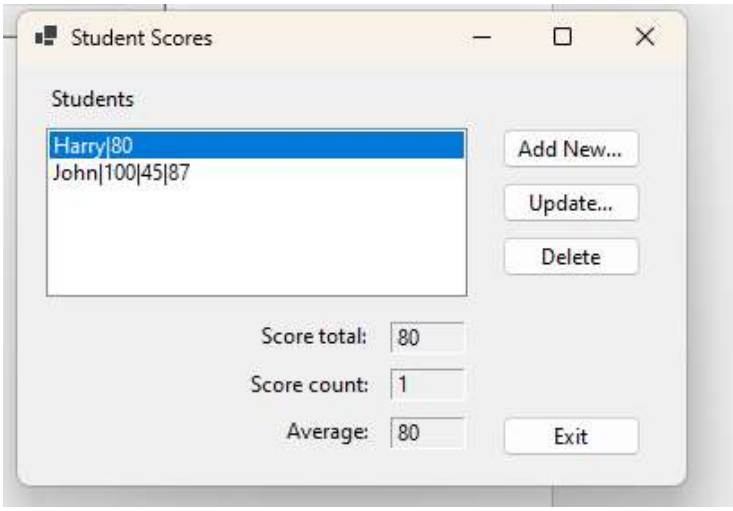

Student Scores
Students
Harry|80
John|100|45|63|87
Shallu|90
Add New...
Update...
Delete
Score total: 80
Score count: 1
Average: 80
Exit

Let’s remove 63 score for John.

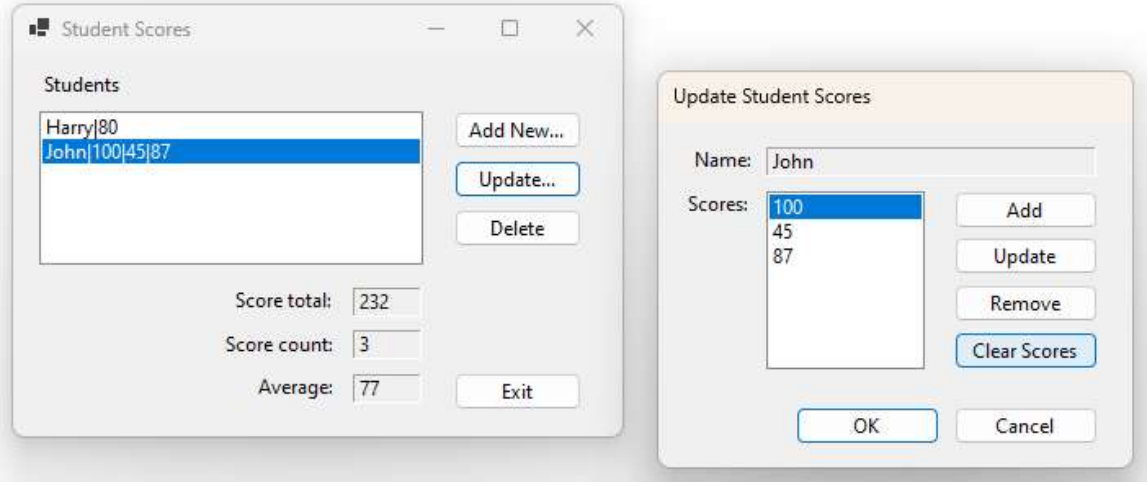


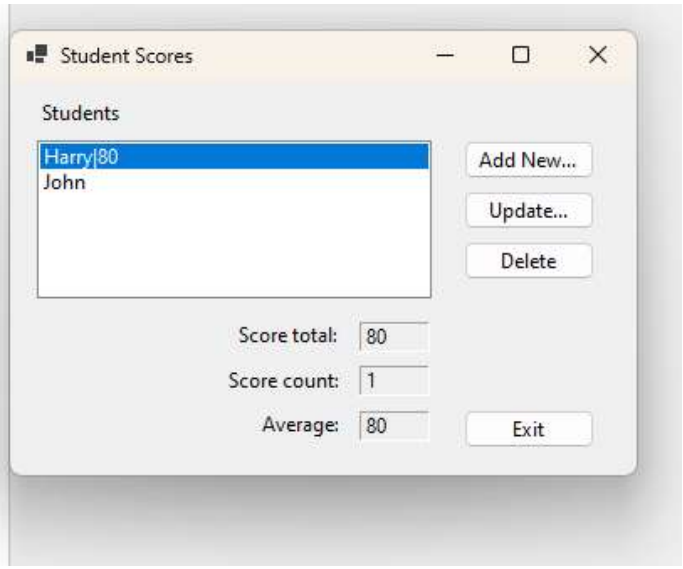


Delete Student Shallu



Let's clear all scores for John.





Deliverable:

1. C# multiform desktop application.

Submission:

1. A compressed zip folder of C# solution to canvas Assign 06 folder.
2. In a separate Word document, include a retrospective that discusses:
    a. What went well with the assignment?
    b. What did not go well -- what did you struggle with?

Note:

1. The assignment awards 20 points according to the criteria given above. Partial or incorrect completion of the elements will reduce points awarded.
2. The retrospective will not be graded but it is required. Failure to include a retrospective will result in a 5% reduction in points.
3. Use best practices to write code as discussed during class lectures.