# :

*Overall:*

* Looks good.

*MarsLanderHistory.RoundInfo class:*

* Looks good ( and unchanged ☺ )

*Mars Lander History class:*

* Looks good, overall
* It’s fine if you add instance variables to this class, but you then must update the Clone() method so that it actually copies all the fields over (-4)
* You did not implement the AddRound method (-15)
* AddRound: you need to resize the array when you run out of space. (-10)
* AddRound: you need to resize the array when you run out of space. (-10)  
  Do this yourself (instead of calling the Array.Resize method) so that you’ve got this skill  
  Setting the array size to be really large is a clever way to avoid this, but it's important that you be able to do this if you wanted/needed to.
* AddRound: Refactor the code so that you don’t repeat the lines that actually add the new value to the array (-3)
* AddRound: Resizing the array will be an O(N) operation (where N = the number of elements in the array). If you only increase the size of the array by one each time (once you’ve reached the limit) this means that your program will have to endure an O(N) operation each round. While the actual numbers are small enough that this isn’t going be a terrible ordeal in practice it’s good to look for stuff like this and fix it. So instead of increasing the size by 1 each time, increase it by (say) 10 or 20 (-6)
* You did not implement the NumberOfRounds method (-6)
* You did not implement the GetHeight method (-6)
* You did not implement the GetSpeed method (-6)
* The parameter should be which round to get the height/speed for the lander (i.e., GetHeight(0) is the height at the end of the zeroth round (i.e., after making a single 'move' in the game), GetHeight(1) is the height after the second time the user burned some fuel, etc, etc) (-3)
* You can (and should) implement the Get\_\_\_ methods in one line (-3)
* This is missing entirely. (-30)

*Mars Lander class:*

* Looks good, overall
* The player doesn't start out with 500 fuel points. (-6)
* The lander doesn't start out 1,000 meters above the surface of the planet. (-6)
* As per the example game, the ship starts out with a downward velocity of 100 m/s. (-6)
* Remove the ‘lander’ part of the variable names – you know it’s part of a lander because it’s an instance variable in the MarsLander class (-3)
* You did not implement the Get/SetSpeed method (-9)
* You did not implement the Get/SetFuelPoints method (-9)
* You did not implement the Get/SetHeight method (-9)
* SetHeight: You let the height go negative (the lander should never go below the surface of Mars. It may end up as a pile of wreckage on the surface of Mars, but it'll still be on the surface. (-6)
* setFuel should actually set the fuel, not decrease the fuel by the parameter’s amount. If you want to keep this logic then rename the method ‘decreaseFuel’ or ‘reduceFuel’ or something else that makes it clear that you’re not setting the fuel, but instead decreasing it (-6)
* You did not implement the CalculateNewSpeed method (-15)
* CalculateNewSpeed: You don’t actually need the “ \* 1” – since there’s no units in C# it doesn’t actually do anything. (-3)
* You don't use the correct calculations when figuring out the new speed. (-6)
* You don't calculate the new speed, THEN have the ship descend that much (i.e., the ship descends at the current speed, THEN changes speed, which is incorrect). (-6)
* You did not implement the GetHistory method (-15)
* If the lander goes below 0m then you let the Lander keep the negative height but add “zero” to the history. This means that the last round of the game you’ll tell the player that the lander is at the negative height, then report zero as the height in the history. Make this consistent so that the lander never goes below zero (-6)
* This is missing entirely. (-40)

*Program class / Program.main method:*

* Looks good, overall
* You never call AddRound anywhere (-9)
* This is missing entirely. (-90)

*UserInterface class:*

* Looks good, overall
* You don't clearly separate the user interaction into a single class. (-40)  
  All the user input (except for printing out data from the lander) should go into the user interface class.
* You did not implement the PrintLocation method (-15)
* PrintLocation: Combine the two separate loops to avoid duplicating code. (-6)
* PrintLocation: Use integer division to get top (instead of converting the integer to a double, passing it to Math.Round, then converting it back to an integer (-6)
* PrintLocation should print at least 1,000m worth of output. (-6)
* PrintLocation: Instead of checking here for negative heights, make sure that the SetHeight method won’t let the altitude to go negative (and then use that method) (-6)
* Your graphical display doesn't properly place the \* - if the ship is between 800 & 899 (for example)(inclusive at both ends), then the \* should go next to the 800 mark. (-6)
* Your code won't print out a picture (with the lander), if the lander is higher than 1000 meters. (-6)  
  (The picture should be extended upwards, so that the \* is drawn at the appropriate height in the picture)
* You do NOT need an array in order to print out the 'graphical depiction' of the height that the lander is currently at, as the game is being played. If your code happened to have such an unneeded array, then you need to remove it (and implement a solution that does not require an array). (-15)  
  (You may need/use an array for other purposes, such as keeping track of data for use at the end of the game)
* You did not implement the PrintLanderInfo method (-15)
* You don't print out the current, exact height speed. (-6)
* You don't print out the current, exact downward speed. (-6)
* You don't print out the current, exact fuel points left. (-6)
* You did not implement the GetFuelToBurn method (-15)
* GetFuelToBurn:Return the amount of fuel to burn but don’t change the lander (yet). This way you can choose what to do with that value (instead of always having to decrease the available fuel) (-6)
* GetFuelToBurn: When the user types in bad input, remind them where the lander is (-6)
* GetFuelToBurn: Refactor the code so that the ‘reminder’ printout isn’t duplicated (-6)
* In GetFuelToBurn, only print the reminder info if the user has typed in a bad input. (-6)
* A player can bid more than their current fuel points (e.g., bidding 600 points at the start) (-6)
* A player can bid a negative number of fuel points (e.g., bidding -60 points) (-6)
* In GetFuelToBurn – you don't provide an error message for when the user types in something that's not a number. (-6)
* Give the user specific feedback on why their fuel-to-burn number was wrong: that they don't have that much fuel, OR that their number is negative, based on what went wrong, instead of both. (-6)
* GetFuelToBurn: If a player attempts an illegal move (bidding negative, or more than they've got), you don't allow them to reenter their bid immediately, instead of cycling through the menu again. (-6)
* If the lander runs out of fuel, the game should continue (with the lander falling each round), until the lander crashes, or lands. It's fine to force the user to type '0' for the amount of fuel they want (-6)
* GetFuelToBurn should keep asking the user for input until they provide valid input (-6)
* You did not implement the PrintEndOfGameResult method (-15)
* PrintEndOfGameResult won’t print anything if the height is negative but the speed is ok (-6)  
  (And your program doesn’t prevent the height of the lander from going negative)
* In PrintEndOfGameResult, use the maxSpeed parameter, instead of assuming that 10 m/s is the maximum safe speed. (This includes printing messages to the user) (-6)
* In PrintEndOfGameResult, use the maxSpeed parameter for the print method as well (-6)
* When the lander reaches (or goes below) 0 meters altitude, you don't properly figure out if it's crashed or not. (-6)
* You did not implement the PrintHistory method (-15)
* This is missing entirely. (-60)

**Grade (out of 130):**

**To calculate your grade:** add up all the (-1)'s and (-6)'s and (-X)'s, to get some negative number, then take that from the total to get your grade. For example: If the total points available for the assignment was 100, and you had the following penalties: -3 + -6 + -6 🡺 -15, so the grade would be 100 -15 = 85.

Why do you have to do this? Because this is only version 1, and so you won't really get your 'real grade' until you hand in the revision. Sometimes the grade on this first version appears really low (especially if you left out a whole section), and so I want to give people feedback, but try to avoid spooking people. Keep in mind that if you don't hand in a revision, this will be your final grade.

**Note**: While the above list of errors is intended to guide your improvement of this homework, you should realize that a given error may occur in more places than have been specifically cited here. It is your responsibility to find and fix all occurrences of a given error, even if they’re not all specifically listed above.

**Note**: Items that are 'greyed out' are there for informational purposes, to preemptively give you feedback for your revision, but these don't actually represent points that you've lost yet. When doing your revision make sure to pay attention to these items so you don’t lose these points in your revision. Example of a 'greyed out' item:

You didn't do X. (-6)