# Assignment 2 (First, initial, not-yet-revised version):

### Upload the INSTRUCTORFEEDBACK file directly to Canvas:

* Thank you for uploading a INSTRUCTORFEEDBACK file directly to Canvas! 😊
* Your INSTRUCTORFEEDBACK file must be uploaded directly to Canvas.
Please DO NOT put your INSTRUCTORFEEDBACK inside the .ZIP file(-5)

## Part 1: Algorithm Implementation:

* Looks good, overall.
* This is identical to the starting project. -130

Did you mean to hand in a different file?

There are no extensions for the initial version of the homework assignments / projects, but you can AND SHOULD hand in something for the final, revised version.
I’d be happy to skim over your work before you hand it in – before/during/after class might be a good time for this. I want to be clear that I never ‘pre-grade’ anything, but skimming over your work is a good way to catch huge errors before you hand this in.

I’m going to leave the following here to help guide your work on the revision
* You used a doubly-linked list, which was not covered in class. It’s great to go looking online for extra help & guidance, but you need to be able to apply that knowledge to the material that we’re covering in class (in part because I want to make sure that you actually understand this stuff, and you’re not just copying-and-pasting from websites).
Redo this using a singly-linked list like what we covered in this class -90
* You need to write the LinkedList class yourself, from scratch – do not use the built-in version that already comes with C# -90
* MyHistoryList and FutureStack are the functionally equivalent (they do the same thing). Remove the (since it doesn’t use a linked list) and then rewrite the history class to use the for both past and future. -10
* Website and FutureWebsite are the exact same classes (except for the name). Get rid of one and then you can use the other node class for both lists. -6
* Make your ‘node’ class into a private, nested class. Then mark the variables as being public and then remove all the get/set methods. -6
* PrintAll is missing and/or does not work -20
* PrintAll doesn’t quite work the way it does in the assignment.
You start printing from the very first item (the oldest item)– the assignment example prints starting from the most recent item and working back to the oldest item. -6
* PrintAll: Refactor the logic that actually prints out the list(s) into a separate method so that you can just call that method (once for each list) instead of duplicating the code in PrintAll
 -6
* You don’t need a separate case for adding to the front of the list when it’s empty, and then adding to the list when it’s not empty. You can add to either list in about 4-5 lines of code -6
* MoveBackwards is missing and/or does not work -20
* MoveBackwards: This will crash if the list is empty -6
* MoveBackwards: Remove the item from the front of this list, and then add it to the ‘future’ list. You’re not adding it to the ‘future’ list -6
* MoveBackwards: You remove the current node, take the string out, and then create a brand-new node to store the string that you add to the future list.
Change this so that you just add the same node to the future list, instead of creating a new one
Fix similar problems elsewhere in the code. -6
* MoveBackwards: Rewrite the ‘add to front’ logic so that it doesn’t matter if first is null or not. You can remove the if/else, replacing that with only the code from the else, and have it work just fine.
Make this change AND put in a quick (1-2 sentence) comment explaining why this works both when the first reference is null AND when it’s not null
Make this change throughout your program. (You don’t need to copy the comment, but you do need to make the change everywhere.) -6
* MoveForwards/Backwards: If you’re already at the end of the future/history list and you call these then you’ll add a ‘null’ string to the other list -6
* MoveForwards is missing and/or does not work -20
* VisitPage is missing and/or does not work
You should move the current page to the front of the ‘history’ list, then remove all future items, then add the new item as the current page -20
* VisitPage: Use a constructor on the ‘node’ class so that you can create the objects in a single line -6
* VisitPage: you must also remove all the entries in the ‘future’ list -6
* VisitPage: you must also remove all the entries in the ‘future’ list
First put in a comment stating (using the Big Oh notation) the running time of your current solution. Then go back and change the code that it takes constant time ( O(1) ) to clear the future list -6
* MoveBackwards, MoveForwards, VisitPage, Delete, etc:
These aren’t supposed to print anything – only PrintAll interacts with the user -6
* Remove unused / commented out code -6
* Looks good, overall

# Grade **(**out of 130**):**

In order to calculate your grade, you need to do the following: add up all the numbers in the right-hand margin (example: -2 + -2 + -3 = -7). Add that number to 130 (example: 130 + -7 = 123)

Note: If you have any duplicate errors, you may see that the second error is listed as "-0". This means that you don't lose any more points for the second time you made the same mistake. You still have to fix all the mistakes.

You also still have to fix all the mistakes even if I don't specifically mention it the second (or third, etc) time. (If you don't have any duplicate errors, you can just ignore this 😊 )

Note Also: If you see some items that are formatted like so:

* You've put down a redundant wall. (-2)

It means that you are NOT losing those points on this version, but if/when you hand in the revision, you should be extra-careful not to make that particular mistake.