WinForms “Notepad” Application / Fibonacci BigInt Text Reader

Cpt S 321 Homework Assignment

Washington State University

Submission Instructions:

Check your program and the rest of the project materials (.vsproj, .sln files and such) into your git repo. Your project should be in a directory at the root of your git repo, the code checked in and pushed to the git server, and the commit you want graded should be tagged. To make Blackboard happy, please upload a small file of your choice to the assignment there so the TA knows you’ve completed the work.

Project Directory name:	NotepadApp
Git tag for turnin:	NotepadApp-v1.0

Assignment Instructions:

Read all the instructions *carefully* before you write any code.

Using Visual Studio or SharpDevelop, create a WinForms application that can load and save plain text files. This will be much like a simplified version of the Notepad utility application that comes with Microsoft Windows. This app will also include the ability to load text from a non-file source (Fibonacci in-memory sequence generator).

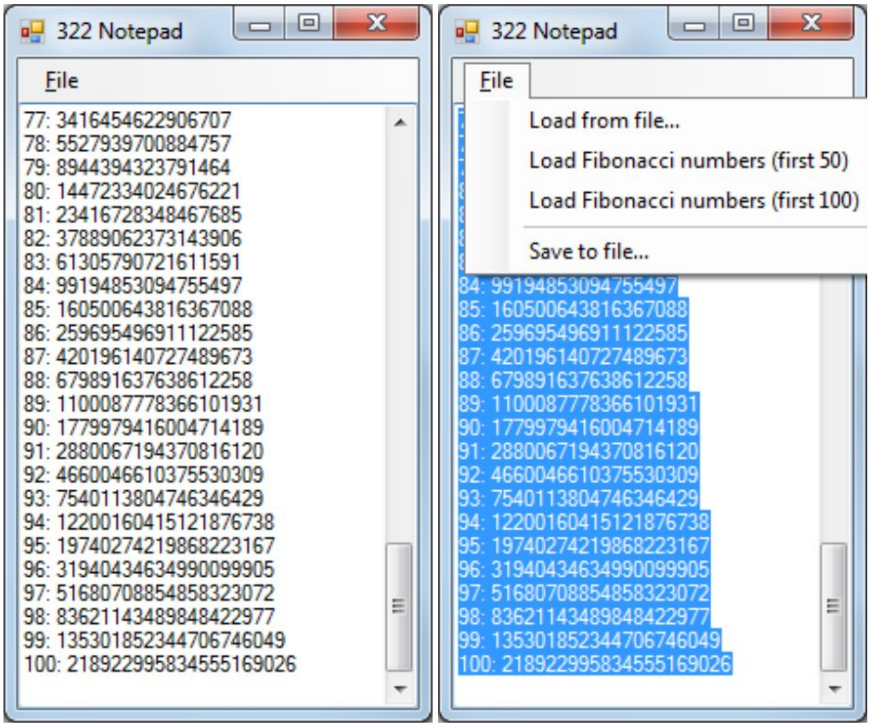
Interface:

You are free to design the interface the way you wish for the most part, but you must include the following:

- a prominent text area that contains the text
    - Use a TextBox control
    - Set the Multiline property to true
    - Set the ScrollBars property to Vertical or Both so that vertical scrolling can occur
- buttons or menu options to load and save files
    - You will need 3 options for loading text:
        - one from file
        - two from another source.

  - This is discussed more below.
- use OpenFileDialog/SaveFileDialog for selecting files to open/save
  - Can be created in code or at design time by dragging them in from the toolbox

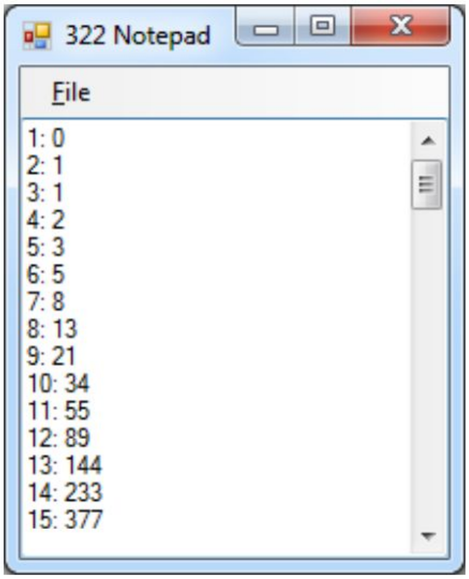
The image below shows an example of what the interface could look like:


322 Notepad
File
322 Notepad
File
Load from file...
Load Fibonacci numbers (first 50)
Load Fibonacci numbers (first 100)
Save to file...

```
77: 3416454622906707
78: 5527939700884757
79: 8944394323791464
80: 14472334024676221
81: 23416728348467685
82: 37889062373143906
83: 61305790721611591
84: 99194853094755497
85: 160500643816367088
86: 259695496911122585
87: 420196140727489673
88: 679891637638612258
89: 1100087778366101931
90: 1779979416004714189
91: 2880067194370816120
92: 4660046610375530309
93: 7540113804746346429
94: 12200160415121876738
95: 19740274219868223167
96: 31940434634990099905
97: 51680708854858323072
98: 83621143489848422977
99: 135301852344706746049
100: 218922995834555169026
```

```
84: 99194853094755497
85: 160500643816367088
86: 259695496911122585
87: 420196140727489673
88: 679891637638612258
89: 1100087778366101931
90: 1779979416004714189
91: 2880067194370816120
92: 4660046610375530309
93: 7540113804746346429
94: 12200160415121876738
95: 19740274219868223167
96: 31940434634990099905
97: 51680708854858323072
98: 83621143489848422977
99: 135301852344706746049
100: 218922995834555169026
```

Here’s another image of the application scrolled up to the beginning numbers:



```
1: 0
2: 1
3: 1
4: 2
5: 3
6: 5
7: 8
8: 13
9: 21
10: 34
11: 55
12: 89
13: 144
14: 233
15: 377
```

Saving Functionality (3 points):

Your application needs to provide the ability to save the text to a text file. Provide a button or menu option that brings up a SaveFileDialog, prompting the user for a file name. If the user clicks OK in that dialog, which you'll determine by checking the return value of the SaveFileDialog.ShowDialog function, then save the text in the text box to that file. You can save it using streams, text-writers, or any other method you see fit (it's the loading that's going to be more specific).

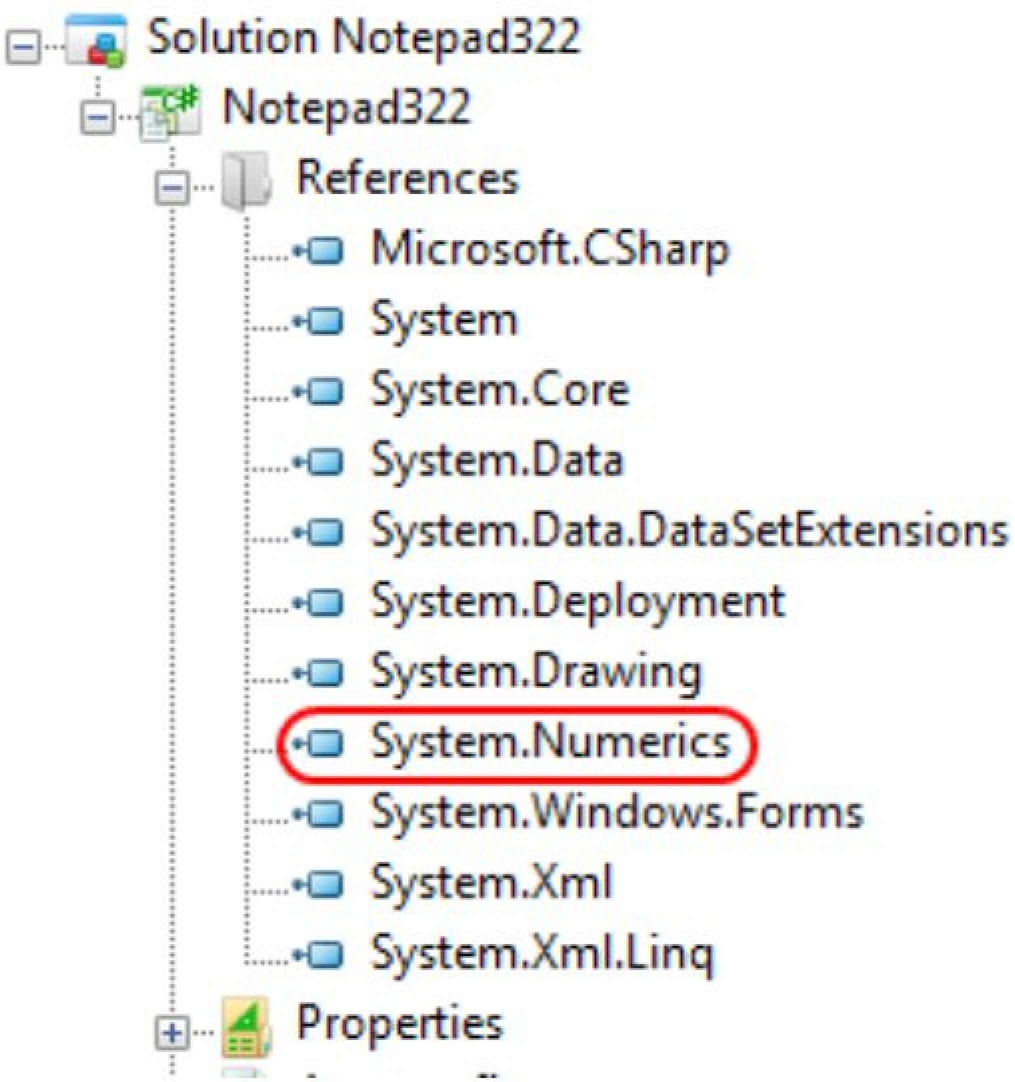
Loading Functionality (10 points total):

1. Generic Loading Function (2 points)
    a. Write a function called "Load" or "LoadText" in your main form's .CS file. This will take a System.IO.TextReader object as a parameter. It should read all the text from the TextReader object and put it in the text box in your interface.
    b. Declare and implement the method mentioned above:

        private void LoadText(TextReader sr)

    c. Use either ReadToEnd or ReadLine (in a loop) to load all text from the reader and put it in the text box. This should replace any content already in the text box.
2. Loading from file (2 points)
   Add the option to load text from a file. You can use a stream reader to open the file and pass it to your loading function. Remember that StreamReader inherits from TextReader and has a constructor that takes a file name as a parameter.
3. Loading from FibonacciTextReader (6 points)
   Write a class named FibonacciTextReader that inherits from the System.IO.TextReader. Since you've implemented a loading function in part 1 that takes a text reader, you'll be able to pass this object to your loading function after creating an instance of it. This class shall:
    a. Inherit from the TextReader class
    b. Make it have a constructor that takes an integer as a parameter indicating the maximum number of lines available (the maximum number of numbers in the sequence that you can generate).
    c. Override the ReadLine method which delivers the next number (as a string) in the Fibonaci sequence. You need logic in this function to handle the first two numbers as special cases. You must return null after the nth call, where n is the integer value passed to the constructor.
    d. Override the ReadToEnd method if you used it back in the implementation for part 1. Implement it such that it repeatedly calls ReadLine and concatenates all the lines together. You can use the System.Text.StringBuilder class to append the lines together into one string.
    e. Use the System.Numerics.BigInteger struct for the sequence numbers. The numbers get beyond what a 64-bit integer can store very quickly, so you need this to accurately compute the first 100 numbers in the sequence. You may have

to add a reference to the System.Numerics DLL in your project for this. Right-click references -> Add reference -> find System.Numerics.dll.



f. Optional: put the line number before the sequence number as shown in the sample app screenshot.

4. Add three menu options (or buttons if you want) in your interface to allow for demonstration of the above functionality
   a. There needs to be an option to load from a file. This should prompt the user for a file name (use OpenFileDialog for convenience) and then load it as described in #2 above.
   b. There needs to be an option to load the first 50 numbers of the Fibonacci sequence
   c. There needs to be an option to load the first 100 numbers of the Fibonacci sequence
   d. (note: parts b and c should have almost identical code, just with a different number passed to the FibonacciTextReader constructor)

5. Put your name and ID number in a comment and the top of each code file that you write. Put a reasonable number of comments in your code.

Scoring breakdown:

- This assignment is worth 15 points
    - 3 is for correct saving functionality
    - 10 are for correct loading function
    - 2 is for the remainder of items, such as adequate comments in the code, proper interface design, and so on.