Programming Assignment 4
Wurd

Time due: 11:00 PM Thursday, March 11

The Project 4 specification document is posted.

Updates:

- 3/10 11:00 pm: If when you run the sample Linux executable or build and run your program on cs32.seas.ucla.edu you get three error lines starting with this:
  `./wurd: /lib64/libstdc++.so.6: version GLIBCXX_3.4.20' not found (required by ./wurd)`
  then download the new Linux sample executable and the new Linux skeleton, which fix that issue.
- 3/7 10:00 pm: The sample executables for macOS and Linux have been updated to perform as the spec requires for batched undos involving space characters and to fix a bug in the case of loading an empty file and then trying to edit it. Windows people can try the Linux version on the SEASnet Linux server until the corrected Windows version is available.
- 3/7 4:00 am: The Xcode debugger seems to be occasionally causing a sequence of a few spurious characters to be inserted into the text when you're debugging. The way to stop that is simple, and it's harmless even if you're not debugging or not even using Xcode: After line 121 in EditorGui.h, add
  ```
  case -1:
      return true;
  ```
- 3/6 11:20 pm: We believe the remaining problem with very long path names to dictionary or text files to edit has been solved. The skeletons have been updated.
- 3/6 8:15 pm: On spec p. 11, the runtime requirement for getlines was revised to include a term for the original size of the lines vector parameter. On spec p. 12, the runtime requirement for undo was revised to include a term for the distance from the current cursor position to the position of the undo operation.
- 3/6 4:45 pm: You can now debug using Xcode! Also, most of the remaining issues some people were having with the previous skeleton have been resolved in the revised Mac (Xcode) skeleton. To use the Xcode debugger (try it first on the skeleton after saving your six Student*xxxxx* files in a safe place):
  1. Start the program and let it get to the point where a Terminal window is open and Wurd is waiting for you to type something. (It's paused in TextIO::getChar waiting for user input.)
  2. Select Debug / Detach from Wurd.
  3. Select Debug / Attach to Process, and of the two Wurd processes shown in the pop up, select the one with the larger number in parentheses (it will usually be the second one).
  4. To check out that it works, set a breakpoint at line 34 in the skeleton's StudentTextEditor.cpp (the body of the move method).
  5. Activate the Wurd window and press an arrow key. This will cause StudentTextEditor::move to be called, so execution will stop at the breakpoint. The Debug area shows the parameter dir's value to be the enumerated type constant UP, DOWN, LEFT, or RIGHT.
- 3/6 3:00 am: Whoops! One wrong file that was picked up for the Xcode skeleton posted at 12:45 am has been replaced with the right one in the skeleton. If you just want that one changed file, it's this main.cpp.
- 3/6 12:45 am: A new Mac (Xcode) skeleton has been posted that we believe will solve most of the issues people have been having with the previous skeleton. Thanks to some annoying decisions Apple made (e.g., having the Xcode console not act like a normal terminal, limiting ways to pass arguments to an app, etc.) we were unable to meet our goal of letting you use the debugger with your program. What you can do, though, is this:
  1. Save the six Student*xxxxx* files you've been working on in a safe place.
  2. Unzip the skeleton file and build it. When you launch it from Xcode, before it appears, you may get a scary dialog box saying *"Wurd" wants access to control "Terminal.app". Allowing control will provide access to documents and data in "Terminal.app", and to perform actions within that app*. Go ahead and click OK.
  3. A Terminal window should open up with a black background. It may not be the top window on your screen; if you don't see it, perhaps it's behind your Xcode window. At the bottom, it will say that it can't load dictionary.txt, because it's a skeleton file that doesn't implement dictionary loading. It won't do much of anything, so type ctrl-X to quit the program, and y to confirm. A couple of seconds after that, the Terminal window will close by itself.
  4. Replace the six skeleton Student*xxxxx* with the files you saved away. Rebuild the program, and you're off and running again.
  5. Every time you make a change to your source file and rebuild your program, you'll get the scary dialog box. Whenever you re-run your program without having made a change that required rebuilding, it won't appear.
  6. Avoid closing the terminal other than by typing ctrl-X to your program. It sometimes causes Wurd to go into an infinite loop. It may be helpful to have the Activity Monitor utility (in /Applications/Utilities) open so if you hear your fan start running, you can quickly kill the Wurd process.
  7. You must run the Wurd executable built by Xcode from Xcode; you can't double-click on it and have it work. If you want such an executable, build it starting from the the Mac (command line) skeleton.
  8. Unfortunately, you can't use the debugger with the Wurd executable built by Xcode.
  9. The executable's current directory is the one with the source files, dictonary.txt, and several text files you can try editing. Because they're in that current directory you don't need to give a full pathname to those files or other text files you may put there for testing purposes, which is convenient.
- 3/5 10:50 am: TextEdit.h and StudentTextEdit.h in the skeletons were updated to reflect that the spec had always required that the getLines method (pp. 10-11) returns an int (although it hadn't previously said what value that int should have). The void return has been changed to int.

Here are skeletons for Windows, Mac (Xcode), Mac (command line), and Linux. If you look at main.cpp, you'll see the four lines that you are allowed to modify in main.cpp (even though you won't be turning it in) to customize the default path to a dictionary file and the color scheme, in case you find dark red on black to be hard to read.

You don't even need the editor framework for working on StudentUndo and StudentSpellCheck, which are independent of each other. For StudentSpellCheck for example, just take the StudentSpellCheck.h and StudentSpellCheck.cpp files from any skeleton, write a tester.cpp file with a main routine that creates a StudentSpellCheck object, call various member functions on it, and verify they behave as they should. The spec points you to our File I/O writeup and a presentation about the trie data structure.

Unzip the sample zip file for Windows, macOS, or Linux, change into the Wurd directory, and read the README file for information on running the sample executable.

Here are the commands you can give to Wurd:

Up, Down, Left, Right arrows
: Move the cursor

Home (Fn+Left on a Mac)
: Move to start of the current line

End (Fn+Right on a Mac)
: End of the current line

PgUp (Fn+Up on a Mac)
: Up one page

PgDown (Fn+Down on a Mac):
: Down one page

Delete (Fn+Delete on a Mac)
: Delete the character under the cursor

Backspace (Delete on a Mac)
: Backspace over the previous character

Ctrl-Z
: Undo last change

Ctrl-S
: Save current the file

Ctrl-L
: Load a new file (on success, discard any changes to the current file)

Ctrl-D
: Load a new dictionary (on success, discard the entire old dictionary)

Ctrl-X
: Exit the editor (discard any changes to the current file)