|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  |  | [ [Home](http://people.uncw.edu/cferner/index.html) ][ [Classes](http://people.uncw.edu/cferner/Classes/ClassesMain.html) ][ [Research](http://people.uncw.edu/cferner/Research/ResearchMain.html) ][ [Links](http://people.uncw.edu/cferner/LinksMain.html) ][ [Biography](http://people.uncw.edu/cferner/BioMain.html) ] |  |
| [Up](http://people.uncw.edu/cferner/Classes/csc434/Assignments.html)  [CS Dept.](http://www.uncw.edu/csc/)     [http://people.uncw.edu/cferner/images/glogo50.gif](http://www.uncw.edu/) |  | **CSC 434**  **Assignment 3 (SML) - Due 3/28/2017**  In this assignment, you will implement merge sort in Standard ML (SML).  You should use the SML New Jersey, which you can get from [www.smlnj.org](http://www.smlnj.org/).  You can experiment and debug using the interpreter, but you can also put your functions and code in a file and load it into the interpreter using the command **use "filename";** .  When you are ready to quit the interpreter, you can type **CTRL-D**, which is the end of file command.  Don't type that twice, or you will get logged out of your command interpreter.  To implement merge sort, you will need the following functions:  **listOfLists:**  This is a *recursive* function the will take a list of values and return a list of lists of those values.  For example:  listOfLists([24,39,50,28,50,1,15,1,35,40,9,8,25,1,2]) returns [[24],[39],[50],[28],[50],[1],[15],[1],[35],[40],[9],[8],[25],[1],[2]]  **merge:**  This is a *recursive* function that will take two lists are parameters, each of which is already sorted, and merges the two lists into one sorted list.  For example:  merge([1,1,9,28],[1,2,8,15,24,39,50]) returns [1,1,1,2,8,9,15,24,28,39,50]  Notice that the lists can be different lengths.  You should also keep the values stable.  In other words, if there are two equal values, they should stay in the same order and equal values from the first list should come before equal values in the 2nd list.  **mergePass:**  This is a *recursive* function that will take a list of lists and merge pairs of lists.  The number of sublists may not necessarily be even.  For example:  mergePass([[24],[39],[50],[28],[50],[1],[15],[1],[35],[40],[9],[8],[25],[1],[2]])returns [[24,39],[28,50],[1,50],[1,15],[35,40],[8,9],[1,25],[2]] mergePass([[24,39],[28,50],[1,50],[1,15],[35,40],[8,9],[1,25],[2]]) returns [[24,28,39,50],[1,1,15,50],[8,9,35,40],[1,2,25]]  **mergeSort:**  This is a *recursive* function that will take a list of lists and repeatedly calls **mergePass** until there is only one sub list, then returns the sub list (not a list of a list).  For example:  mergeSort([[24],[39],[50],[28],[50],[1],[15],[1],[35],[40],[9],[8],[25],[1],[2]])returns [1,1,1,2,8,9,15,24,25,28,35,39,40,50,50].  **mergeSortDriver:**  This is a ***non****-recursive* function that returns mergeSort of listOfLists.  For example:  mergeSortDriver([24,39,50,28,50,1,15,1,35,40,9,8,25,1,2]) returns [1,1,1,2,8,9,15,24,25,28,35,39,40,50,50]  After you have all this tested and working, you can then use **mergeSortDriver** to sort a list of values.  **Input/Output:**  You should have a main program to creates a list (at least 50) of random values between a min and max.  It should display/print the unsorted list.  It should then use the mergeSortDriver to sort the list and display/print the sorted list.  **Documentation:**  It is expected that you will follow standard practices of documentation of your program.  (Comments in SML are placed inside the symbols "**(\***" and "**\*)**".  This can be nested.) That means that functions should have header information include: Author, date written, list and description of parameters or data members (where appropriate), type and description of return values (where appropriate), a general description of the purpose of the function.  You should also put in comments for code for which its meaning is not obvious.  That does not mean to put in comments that are obvious to skilled programmers.  You should also put in comments for constructs that are not common in every language (e.g. use of regular expressions).  And finally, any part of the program that asks the user for input should be proceeded with an appropriate prompt.  An appropriate prompt is one that tells the user what to enter and in what form.  For example, "Enter two floating-point numbers separated by whitespace on one line."  **Deliverables:**   * The source file(s) for the program * Explain the features you used in the program. In particular, explain what the language constructs, libraries, and functions do. * Submit the files to the assignment drop box on Blackboard. |  |
| This page was last updated: **March 23, 2017**  **Email: http://people.uncw.edu/cferner/images/mailme.png**  [ [Home](http://people.uncw.edu/cferner/index.html) ][ [Classes](http://people.uncw.edu/cferner/Classes/ClassesMain.html) ][ [Research](http://people.uncw.edu/cferner/Research/ResearchMain.html) ][ [Links](http://people.uncw.edu/cferner/LinksMain.html) ][ [Biography](http://people.uncw.edu/cferner/BioMain.html) ] | | |  |