The program I wrote is a basic tokenzier which takes a user given argument and tokenzies with all non-letters as delimiters as well as sorting it using insertion sort. The program is written in C and is very thoroughly commented. It contains features such as: malloc(), realloc(), free(), pointers, pointer to pointers, array referencing notation of pointers, modularity via 3 helper functions, an array turned to an arraylist if necessary and other basic programming tools.

The basic structure of the program is as follows: The program accepts 2 argument from the user; The name of the program that is being run and a string. If there are any more or any less the program will inform the user and terminate. If the there are exactly 2 arguments then the bulk of the program will begin. The program starts with a for loop which is used to analyze each char in the user's second argument. Immediately in the loop there is a check for the char being analzyzed to determined if it's a letter or not. If it's not then the program hits a "continue" and the loop iterates to the next char. If it is a letter then it does a few different things based on whether it is the first letter contiguously, last letter contiguously, or a letter in the middle of other letters contiguously. First, there is a check to see if it is the first letter encountered. If it is, then an int variable is assigned to keep that index in memory so it can later be copied from that index to the ending index as that is where a substring we are interested in sorting and printing begins. After that variable is stored, the program then checks the next char of the user's 2nd argument. First, it checks the size of the argument to the index to make sure it doesn't over run the string. If the next index is out of the bounds of the user's string then there can not be any following letters and this string that is currently being analyzed contains letters, should be stored and is the last to be stored. Thus, it is stored by calling createSubstring() (a helper method I made) and stored and the for loop is exited. If the next letter is not the end of the user string, but it's also a letter then the program uses "continue" to analyze the next set of chars. If the next char is instead not a letter then we have reached the end of the current sub-string and createSubstring() is called to store the string and then the program continues.

Since the data structure used to store the strings is an array the amount of total strings must be kept track. This variable helps know when to extend the array and there is an incrementing variable that is multiplied by the original size of the array (100) to accommodate more strings. The variable storing the total amount of strings is also useful as it gives the precise amount, -1, by which to loop through the final array containing all letter based substrings and sort.

The last part of the program is sorting and printing the sort array. A helper function called insertion sort is used. It executes a basic insertion sort based on ascii values. Insertion sort was chosen because it is stable which means if the user's list was, in some way, previously sorted and they wanted to sort based on another criteria while keeping the relative order of the argument they can. It was also chosen because it is simple to write, read and debug for future optimizations. Considerations to the likely size of the arguments were also given and since it's a command line argument the maximum amount of strings that can appear are 1,000,000 which makes the rare reverse ordered O(n^2) rare and not time limiting for most modern computers to handle. For smaller lists like the kind that can be entered in the command line which are probably at least partially stored, insertion sort will be possible faster than O(nlogn) sorts. Inside insertion sort, alast helper method is swap which is a basic function that is used inside of insertion sort to swap elements each time it's necessary. Finally, the array is printed once sorted. The program then returns to main to free all malloc'ed and realloc'ed memory and terminates with a "return 0" to indicate success.