My program in stringsorter uses a very basic sorting algorithm with the help of a size 52 linked list array. This array is similar to a hashtable's principle, in which every string has uses its first character in a linear hash key. Starting from capital A - Z, and then a - z, my program uses a total space of O(52 + n) which is just O(n) and a time complexity of a linear hash sort which is also just O(n).

Here is a simple algorithm of pseudo-code to follow for sorting. Please refer to my actual program's comments for the whole outlier.
EX: ./a.out "abc1a"

To delimit a string first:

1. Read until string hits first character;
2. a is first character, keep reading;
3. c is last character;
4. Cut string "abc" store in node and use insert method;
5. Read next available character start;
6. a is first character, keep reading;
7. a is only character, cut string "a" and store in node and use insert method;

Our resulting method will use these following steps.

1. String abc has a first letter a;
2. a is 27 with our modified ascii value table, so it goes into hashtable[26];
3. Hashtable[26] is currently NULL, therefore our string, (which is currently stored in a node), is now the head of hashtable[26];
4. Insert complete, we move onto the next real word.
5. String a has a first letter a;
6. Repeat step two.
7. Hashtable[26] is now not null anymore, we use strcmp, and loop until it finds the first word that when compared gives a positive value;
8. However, loop does not even commence, because a > abc, therefore it goes to sort method of linking nodes in the linked list.
9. In this case, because loop did not commence, our new string (inside node) is the new head, so string(node) -> next = hashtable[26]; and then our new hashtable[26] = string(node);

Finally, just print everything from hashtable[0] - hashtable[52].