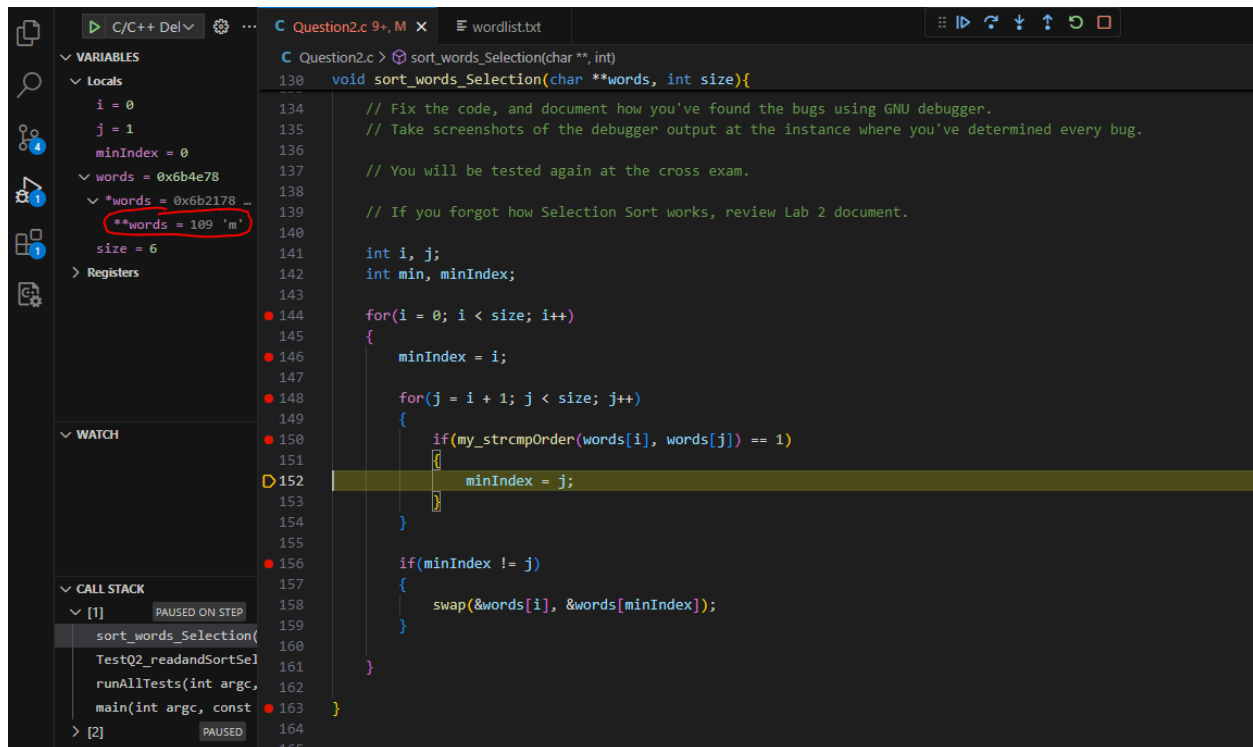


Lab3 Question 2 Debugging Report | xuej41 | 400515671

Bug #1 – Line 150: words[i] should be changed to words[minIndex]

The issue with using words[i] in the inner loop is that it fails to keep track of the smallest element found so far in the unsorted part of the array. In Selection Sort, we need to locate the minimum element in each pass and swap it with the element at the current position, i. To do this, we use minIndex to store the index of the smallest element found as we iterate.

Watch what “m” in milan is replaced by:



The screenshot shows a C++ IDE with a file named 'Question2.c'. The code implements a Selection Sort algorithm. The left sidebar shows the 'VARIABLES' panel with 'minIndex' set to 0 and 'words' pointing to a memory address. The 'WATCH' panel is empty. The 'CALL STACK' panel shows the current function 'sort_words_Selection' and its callers. The main code area shows the function definition. Line 150 is highlighted, showing the condition 'if(my_strcmpOrder(words[i], words[j]) == 1)'. The code is as follows:

```
130 void sort_words_Selection(char **words, int size){
131
132     // Fix the code, and document how you've found the bugs using GNU debugger.
133     // Take screenshots of the debugger output at the instance where you've determined every bug.
134
135     // You will be tested again at the cross exam.
136
137     // If you forgot how Selection Sort works, review Lab 2 document.
138
139     int i, j;
140     int min, minIndex;
141
142     for(i = 0; i < size; i++)
143     {
144         minIndex = i;
145
146         for(j = i + 1; j < size; j++)
147         {
148             if(my_strcmpOrder(words[i], words[j]) == 1)
149             {
150                 minIndex = j;
151             }
152         }
153
154         if(minIndex != i)
155         {
156             swap(&words[i], &words[minIndex]);
157         }
158     }
159 }
```

```
130 void sort_words_Selection(char **words, int size){
134 // Fix the code, and document how you've found the bugs using GNU debugger.
135 // Take screenshots of the debugger output at the instance where you've determined every bug.
136
137 // You will be tested again at the cross exam.
138
139 // If you forgot how Selection Sort works, review Lab 2 document.
140
141 int i, j;
142 int min, minIndex;
143
144 for(i = 0; i < size; i++)
145 {
146     minIndex = i;
147
148     for(j = i + 1; j < size; j++)
149     {
150         if(my_strcmpOrder(words[i], words[j]) == 1)
151         {
152             minIndex = j;
153         }
154     }
155
156     if(minIndex != j)
157     {
158         swap(&words[i], &words[minIndex]);
159     }
160 }
```

As you can see, “m” in milan is replaced by “b” in banana, when it should have been replaced by “a” in apple. This is because using words[i] in the comparison would instead mean we’re always comparing with the element at i, rather than the current smallest element, which prevents the algorithm from correctly identifying the minimum.

By comparing words[minIndex] with words[j], minIndex will point to the smallest word seen so far, and update whenever it finds a smaller element.

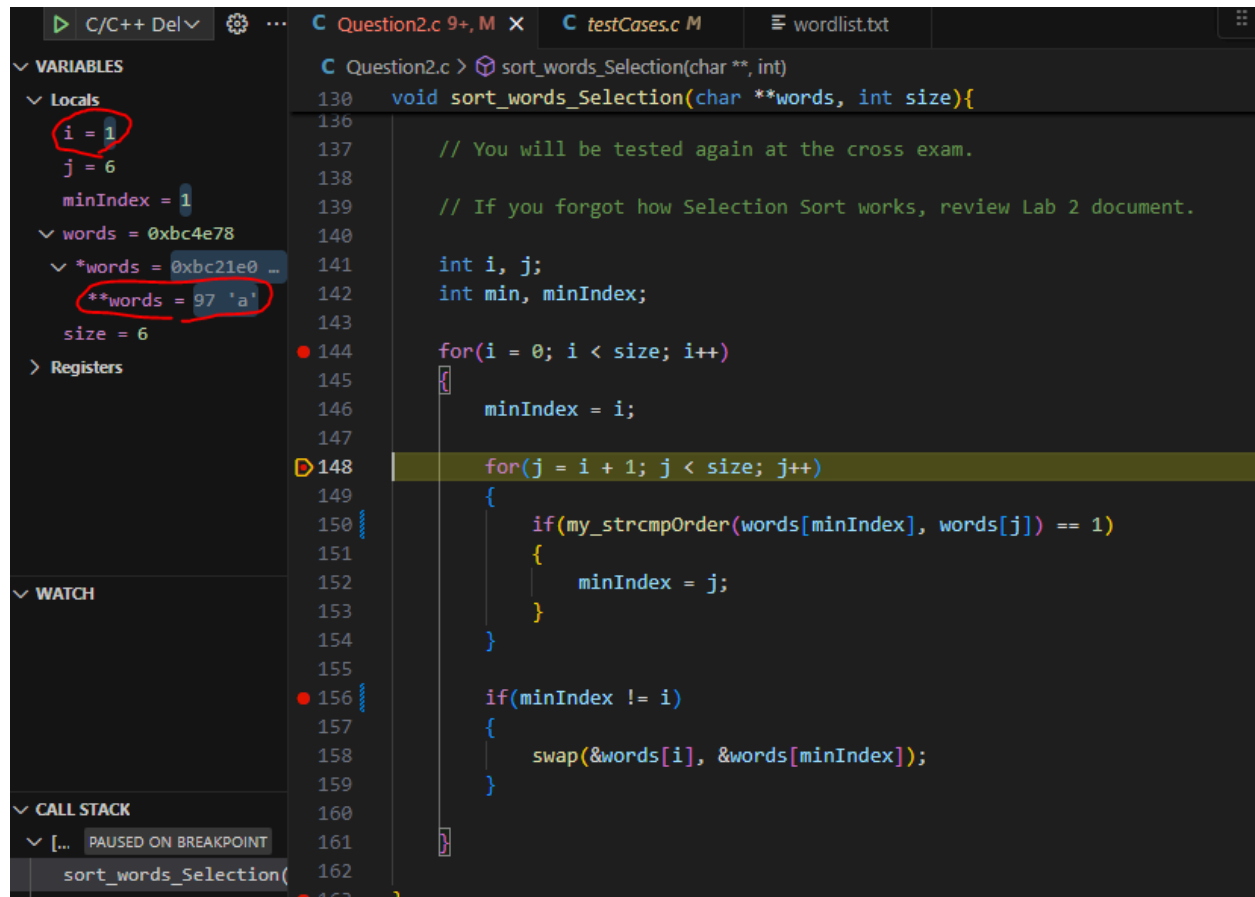
Bug #2 – Line 156: minIndex != j should be changed to minIndex != i

The condition if(minIndex != j) is incorrect because it does not properly check whether a swap is needed. The variable minIndex holds the index of the smallest element found during the inner loop. By the end of the inner loop, we need to check whether this smallest element’s index, minIndex, is different from i, not j, because i is the position we are looking to fill with the smallest value.

Having minIndex != j instead will mean that when the code gets to that line, j = size (as it was incremented to that size in the loop), so having minIndex != size will mess up the logic of the algorithm.

Changing the condition to if (minIndex != i) makes sure to only perform the swap if the smallest element is not already in the correct place.

After implementing these changes, you can see that the algorithm is running correctly. After 1 pass of the sorting algorithm ($i = 1$), "m" in milan has been correctly replaced by "a" in apple.



The screenshot shows a C++ IDE with a debugger window on the left and a code editor on the right. The debugger window displays the following variables:

- VARIABLES**
 - Locals**
 - `i = 1` (highlighted with a red circle)
 - `j = 6`
 - `minIndex = 1`
 - `words = 0xbc4e78`
 - `*words = 0xbc21e0`
 - `**words = 97 'a'` (highlighted with a red circle)
 - `size = 6`
- Registers**

The code editor shows the following code:

```
130 void sort_words_Selection(char **words, int size){
136
137     // You will be tested again at the cross exam.
138     // If you forgot how Selection Sort works, review Lab 2 document.
139
140     int i, j;
141     int min, minIndex;
142
143     for(i = 0; i < size; i++)
144     {
145         minIndex = i;
146
147         for(j = i + 1; j < size; j++)
148         {
149             if(my_strcmpOrder(words[minIndex], words[j]) == 1)
150             {
151                 minIndex = j;
152             }
153         }
154
155         if(minIndex != i)
156         {
157             swap(&words[i], &words[minIndex]);
158         }
159     }
160
161 }
162
163 }
```