Jeremy Scheuerman

COSC 220

Dr. Wang

6 April 2021

Lab 10 writeup

1. The advantage of a linear search is that it is simple
2. The disadvantage of a linear search is that it is slow/time consuming
3. The advantage of a binary search over a linear search us that a binary search is that it is faster/more efficient
4. An advantage of a linear search over a binary search is that the data must be ordered for a binary search
5. After 3 passes of a binary search approximately what fraction of the original array still needs to be searched (assuming the desired data has not been found) ? 1/8
6. While the bubble sort algorithm is conceptually simple it can be inefficient for large arrays because data values only move one at at time.
7. An advantage of the selection sort is that ,for an array of size n, at most n-1 moves are required.
8. Use the bubble sort on the array below and construct the first 3 steps that actually make changes, (Assume the sort is from smallest to largest).

19	-4	91	0	-17
Element 0	Element 1	Element 2	Element 3	Element 4

Element 0	Element 1	Element 2	Element 3	Element 4

Element 0	Element 1	Element 2	Element 3	Element 4

Element 0	Element 1	Element 2	Element 3	Element 4

-4 , 19 , 91, 0 -17

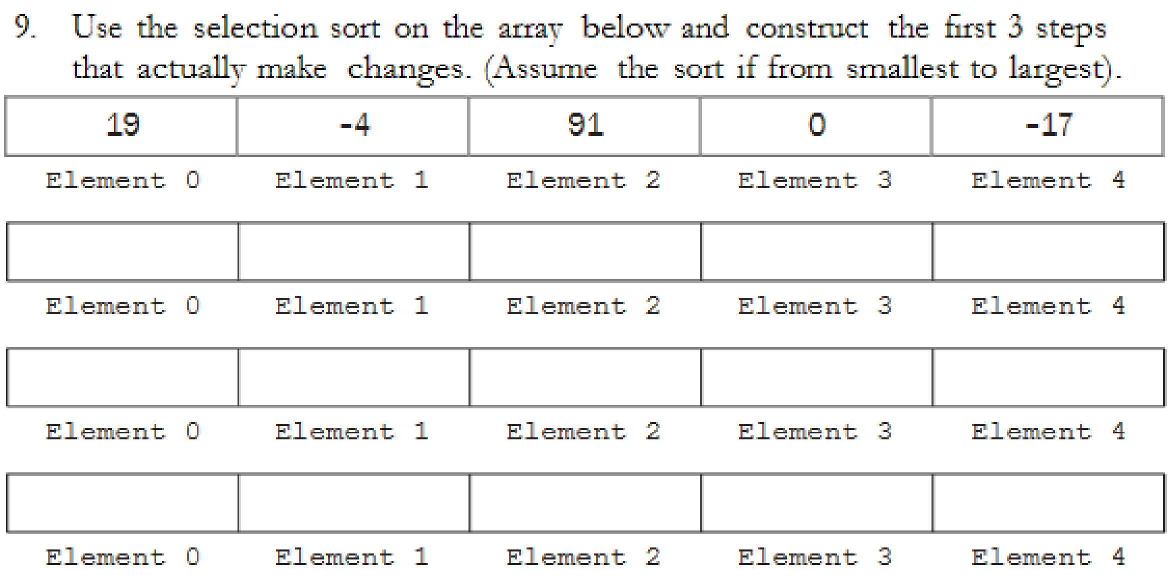
-4, 19, 0, 91 -17

-4, 19, 0, -17 , 91

(now it would repeat the process from the beginning switching the elements)

9. Use the selection sort on the array below and construct the first 3 steps that actually make changes

(Assume the sort if from smallest to largest)



9. Use the selection sort on the array below and construct the first 3 steps that actually make changes. (Assume the sort if from smallest to largest).

19	-4	91	0	-17
Element 0	Element 1	Element 2	Element 3	Element 4
Element 0	Element 1	Element 2	Element 3	Element 4
Element 0	Element 1	Element 2	Element 3	Element 4
Element 0	Element 1	Element 2	Element 3	Element 4

-17, -4, 91, 0, 19

-17 ,-4, 0, 91,19

-17, -4, 0, 19, 91

8.1

8.2

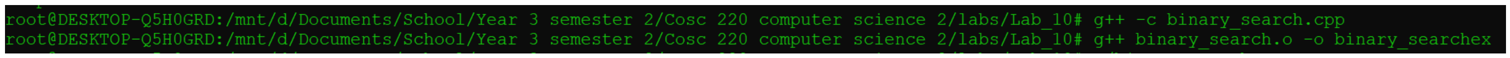
Ex 1

The right side is an integer because if it is run by itself it will round off and stay as an integer

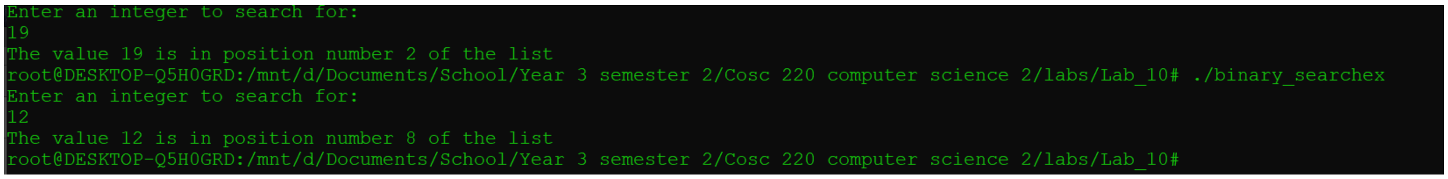
-The middle value is determined by the values of first and last , it is first plus the difference of last and first divided by 2, last will result as an integer (since integer div by integer) so the result will always be an integer

-It affects the program because it can determine where the middle of the array actually is dynamically

Ex 2



```
root@DESKTOP-Q5H0GRD:/mnt/d/Documents/School/Year 3 semester 2/Cosc 220 computer science 2/labs/Lab_10# g++ -c binary_search.cpp
root@DESKTOP-Q5H0GRD:/mnt/d/Documents/School/Year 3 semester 2/Cosc 220 computer science 2/labs/Lab_10# g++ binary_search.o -o binary_searchex
```



```
Enter an integer to search for:
19
The value 19 is in position number 2 of the list
root@DESKTOP-Q5H0GRD:/mnt/d/Documents/School/Year 3 semester 2/Cosc 220 computer science 2/labs/Lab_10# ./binary_searchex
Enter an integer to search for:
12
The value 12 is in position number 8 of the list
root@DESKTOP-Q5H0GRD:/mnt/d/Documents/School/Year 3 semester 2/Cosc 220 computer science 2/labs/Lab_10#
```

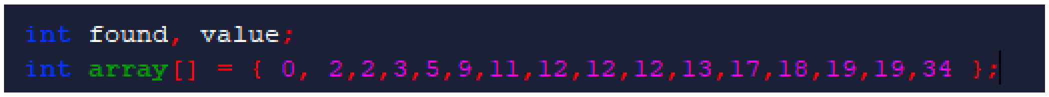
It finds the 1st occurrence of 19

It finds the 3rd occurrence of 12

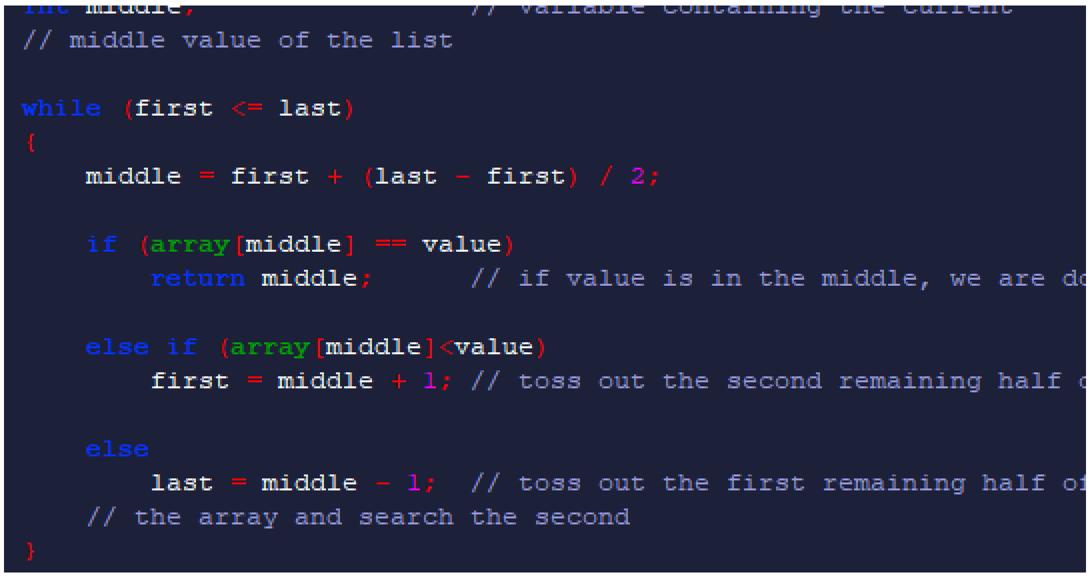
Difference is that we are only looking at the middle point of an upper and lower bound so since the occurrence

happened near the middle point that is the result you get

Ex 3



```
int found, value;
int array[] = { 0, 2,2,3,5,9,11,12,12,12,13,17,18,19,19,34 };
```



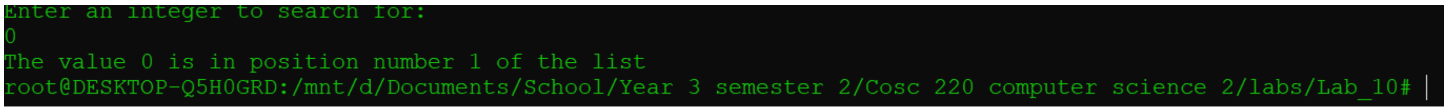
```
int middle;                // variable containing the current
// middle value of the list

while (first <= last)
{
    middle = first + (last - first) / 2;

    if (array[middle] == value)
        return middle;      // if value is in the middle, we are d

    else if (array[middle]<value)
        first = middle + 1; // toss out the second remaining half o

    else
        last = middle - 1;  // toss out the first remaining half of
    // the array and search the second
}
```

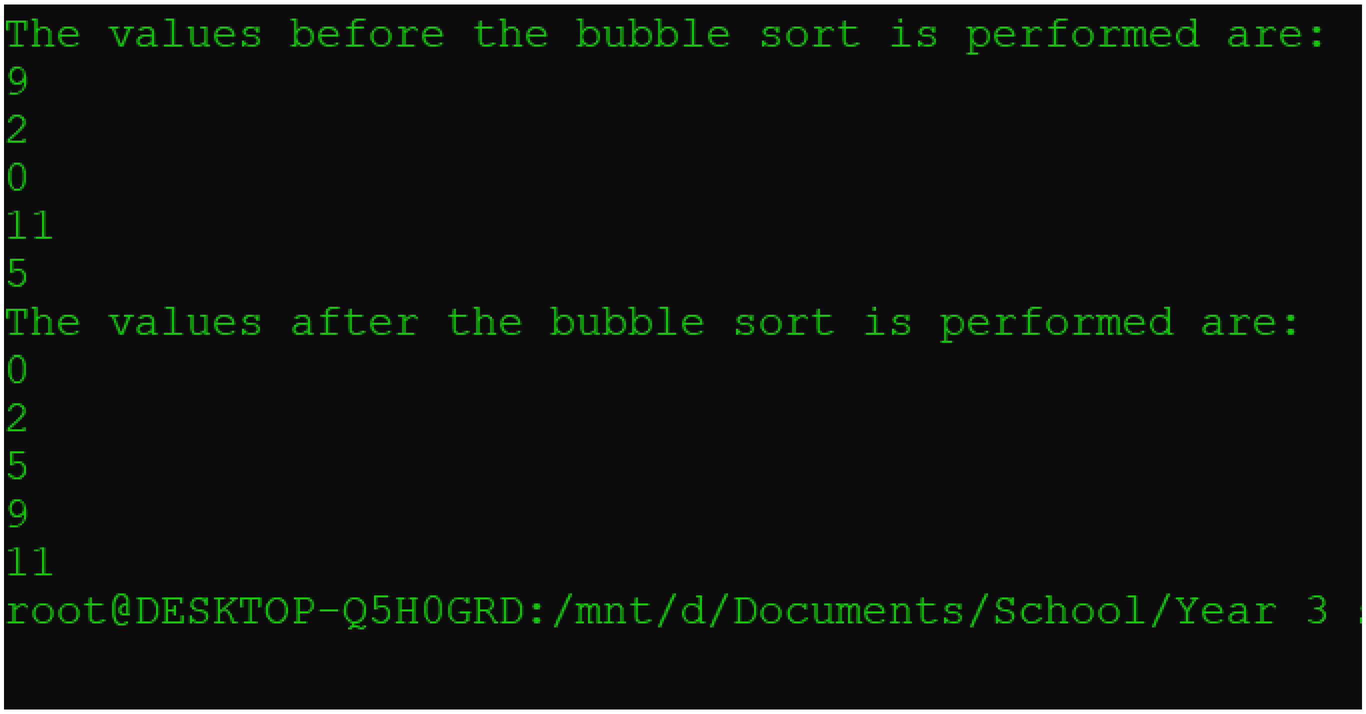


```
Enter an integer to search for:
0
The value 0 is in position number 1 of the list
root@DESKTOP-Q5H0GRD:/mnt/d/Documents/School/Year 3 semester 2/Cosc 220 computer science 2/labs/Lab_10#
```

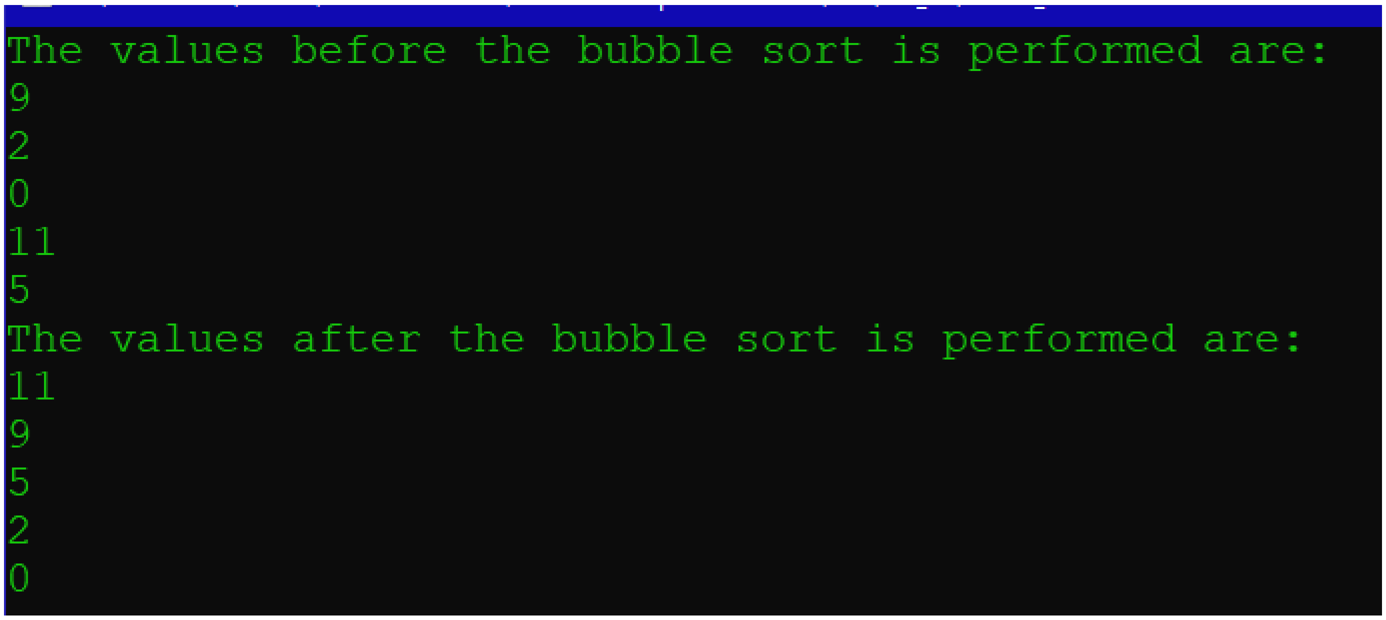
8.3 (with bubble sort)

Ex 1

Original


```
The values before the bubble sort is performed are:
9
2
0
11
5
The values after the bubble sort is performed are:
0
2
5
9
11
root@DESKTOP-Q5H0GRD:/mnt/d/Documents/School/Year 3
```

All you have to do is change the > to a <


```
The values before the bubble sort is performed are:
9
2
0
11
5
The values after the bubble sort is performed are:
11
9
5
2
0
```

No

My Prediction (After the exercise earlier this makes more sense)

9,2,11,0,5

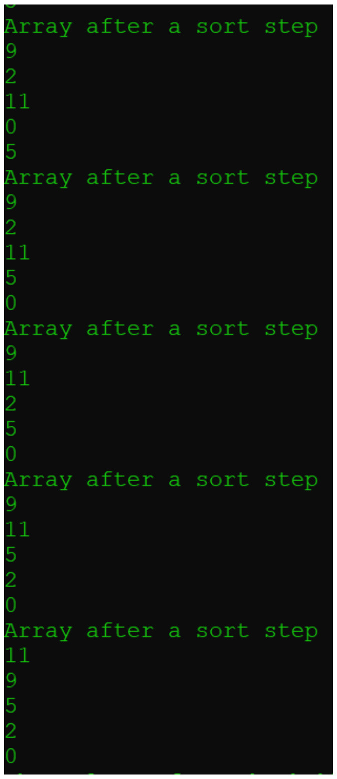
9,2,11,0,5

9,11,2,5,0

9,11,5,2,0

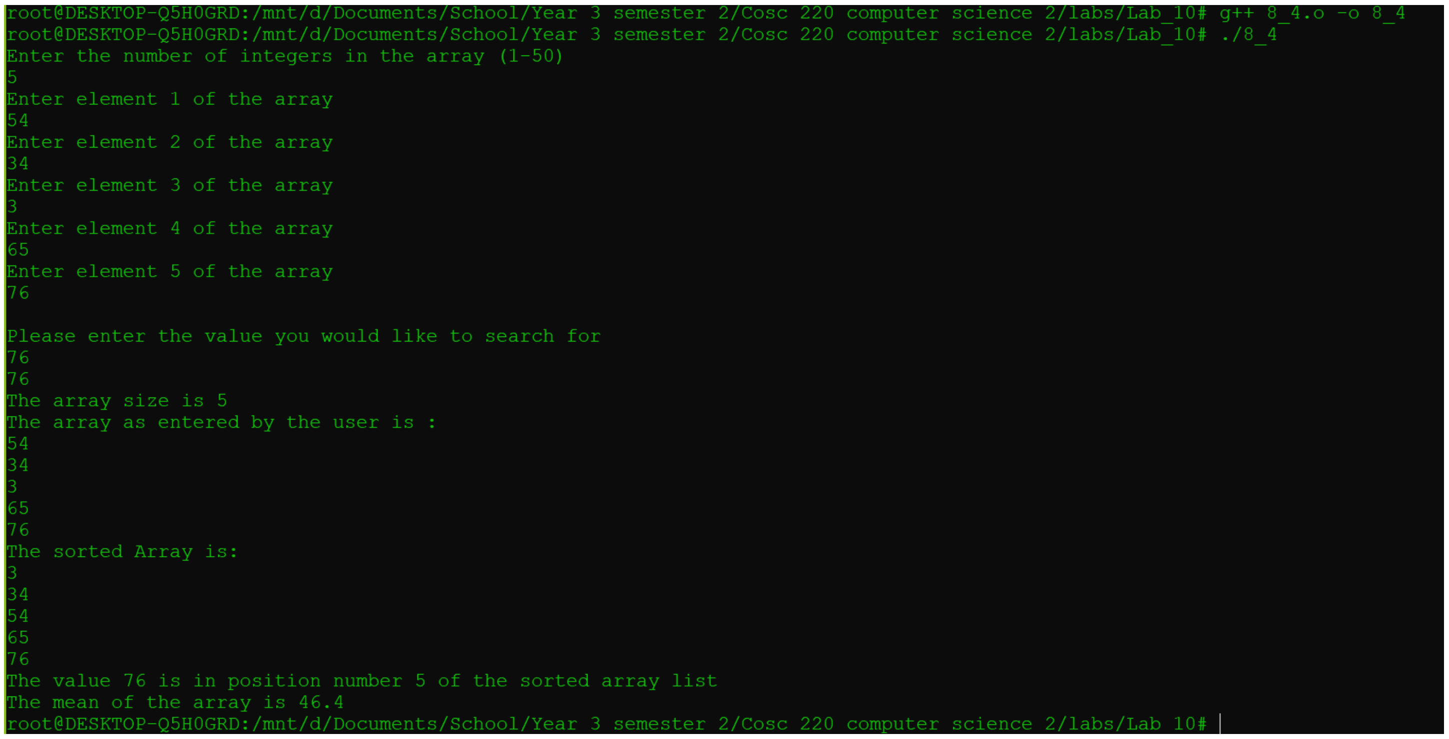
11,9,5,2,0

Actual array after each step



```
Array after a sort step
9
2
11
0
5
Array after a sort step
9
2
11
5
0
Array after a sort step
9
11
2
5
0
Array after a sort step
9
11
5
2
0
Array after a sort step
11
9
5
2
0
```

8.4



```
root@DESKTOP-Q5HOGRD:/mnt/d/Documents/School/Year 3 semester 2/Cosc 220 computer science 2/labs/Lab_10# g++ 8_4.o -o 8_4
root@DESKTOP-Q5HOGRD:/mnt/d/Documents/School/Year 3 semester 2/Cosc 220 computer science 2/labs/Lab_10# ./8_4
Enter the number of integers in the array (1-50)
5
Enter element 1 of the array
54
Enter element 2 of the array
34
Enter element 3 of the array
3
Enter element 4 of the array
65
Enter element 5 of the array
76

Please enter the value you would like to search for
76
76
The array size is 5
The array as entered by the user is :
54
34
3
65
76
The sorted Array is:
3
34
54
65
76
The value 76 is in position number 5 of the sorted array list
The mean of the array is 46.4
root@DESKTOP-Q5HOGRD:/mnt/d/Documents/School/Year 3 semester 2/Cosc 220 computer science 2/labs/Lab_10#
```