

Name: Harris Spahic

Date: 9/21/21

Pledge: I pledge my honor I have abided by the Stevens Honor System.

ERROR 1:

```
Breakpoint 1, contains (arr=0x7fffffffdf10, target=21845) at terrible_dynamic_size_array_unsorted.c:26
26  int contains(struct int_array* arr, int target) {
(gdb) s
30      for (i = 0; i < arr->count; ++i)
(gdb) n
32          if (arr->data[i] == target)
(gdb) n
Ambiguous command "n": macro, maintenance, make, mem, monitor, mt.
(gdb) n
35              return FALSE;
(gdb) n
38      }
```

**Our first error is with the contain function. When we first run out contains function, it appears that the code checks if the target is reached at index 1, then immediately returns false. We want to check EVERY element in the dynamic array, not just the first element so we must remove the (else return false) command from our loop.**

```
int contains(struct int_array* arr, int target) {
    unsigned int i;

    for (i = 0; i < arr->count; ++i)
    {
        if (arr->data[i] == target)
        {
            return TRUE;
        }
        else
        {
            return FALSE;
        }
    }
    return FALSE;
}
```

## ERROR 2:

```
0 0 1431670800 21845 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17
0 0 1431670800 21845 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17 19
0 0 1431670800 21845 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17 19 21
0 0 1431670800 21845 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17 19 21 23
Resize function works properly

Breakpoint 1, contains (arr=0x7fffffffdf30, target=21845) at terrible_dynamic_size_array_unsorted.c:26
26 int contains(struct int_array* arr, int target) {
(gdb) n
30     for (i = 0; i < (arr->count); i++);
(gdb) n
32         if ((arr->data[i]) == target)
(gdb) n
36     return FALSE;
(gdb)
37 }
(i-search)`:
```

Even when we fix the previous extra else case in our “Contains function”, it seems like our code still avoids the loop and goes straight to false. If we look closer we’ll find that some demon added a semicolon before our for loop brackets. Lets get rid of that, and make contain run properly.

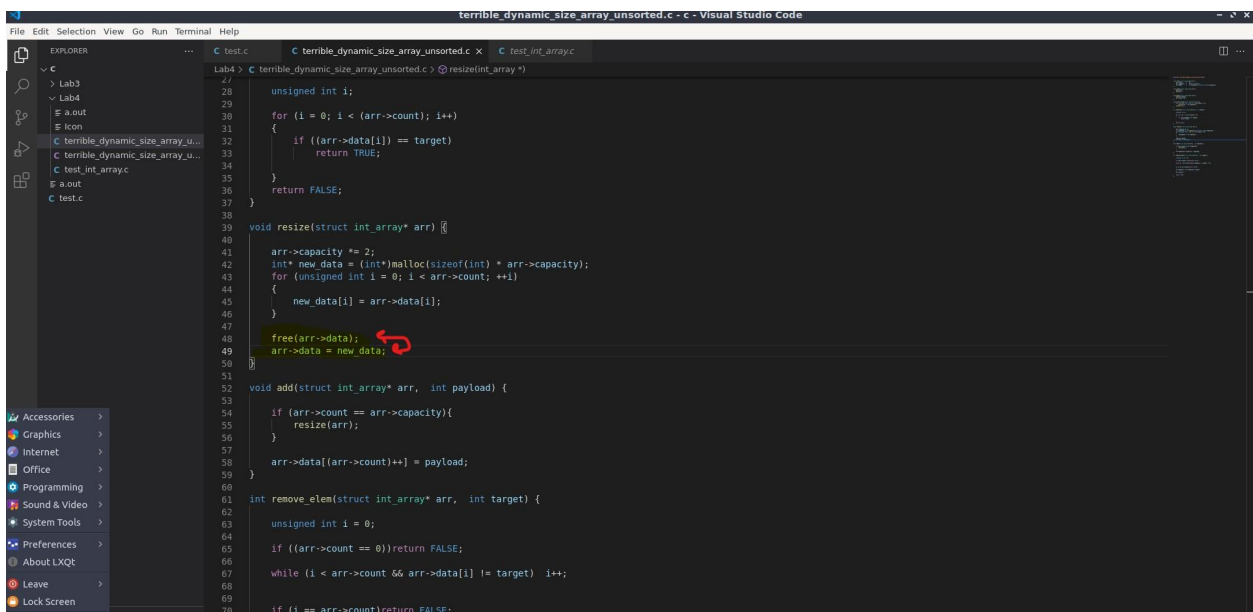
```
24 }
25
26 int contains(struct int_array* arr, int target) {
27     unsigned int i;
28
29     for (i = 0; i < arr->count; ++i; |
30     {
31         if (arr->data[i] == target)
32             return TRUE;
33         else
34             return FALSE;
35     }
36     return FALSE;
37 }
38 }
```

### ERROR 3:

```
44         for (unsigned int i = 0; i < arr->count; ++i)
(gdb) s
49         arr->data = new_data;
(gdb) s
50         free(arr->data);
(gdb) s
0 0 1431670800 21845 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 0
0 0 1431670800 21845 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 0 15
0 0 1431670800 21845 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 0 15 17
0 0 1431670800 21845 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 0 15 17 19
0 0 1431670800 21845 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 0 15 17 19 21
Resize function works properly
Number 6 not in Array Contains not working properly
Number 30 not in Array
Number 23 not in Array error in remove_elem
Number 24 not in Array error in remove_elem
Number 0 not in Array error in remove_elem
Number not in Array
free(): double free detected in tcache 2
```

We have some sort of error with the resizing of the array. From the large random numbers added, we see that this is probably a memory issue. Running line by line in `resize()`, we see that we reassign our data to the `new_data` array; but then free the `new_data` array.

We want to do that in reverse order!



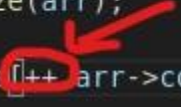
#### ERROR 4:

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...
(gdb) break add
Breakpoint 1 at 0x1689: file terrible_dynamic_size_array_unsorted.c, line 52.
(gdb) run
Starting program: /home/user/Desktop/handout/a.out
Empty array created
Array cleared of data

Breakpoint 1, add (arr=0x7fffffff000, payload=21845) at terrible_dynamic_size_array_unsorted.c:52
52 void add(struct int_array* arr, int payload) {
(gdb) n
54     if ((arr->count == arr->capacity))
(gdb) n
58     arr->data[++ arr->count] = payload;
(gdb) print payload
$1 = 0
(gdb) print arr->count
$2 = 0
(gdb) n
59 }
(gdb) print arr->count
$3 = 1
(gdb) █
```

**Notice that we're using a preincrement for our for loop, instead of a post increment. This will assign our first payload to the second index (1 + 0), second payload to the third index (1 + 1) and so on; shifting our entire array over to the right and cutting off the end. We need to change this to post increment, so our payload is assigned to the first index, and then the index is incremented.**

```
void add(struct int_array* arr, int payload) {
    if ((arr->count == arr->capacity))
        resize(arr);
    arr->data[++ arr->count] = payload;
}
```



## ERROR 5:

```
user@box: ~/c/Lab4
File Actions Edit View Help
user@box: ~/c/Lab4
0 2
0 2 4
0 2 4 6
0 2 4 6 8
0 2 4 6 8 10
0 2 4 6 8 10 12
0 2 4 6 8 10 12 14
0 2 4 6 8 10 12 14 16
0 2 4 6 8 10 12 14 16 18
0 2 4 6 8 10 12 14 16 18 20
0 2 4 6 8 10 12 14 16 18 20 22
0 2 4 6 8 10 12 14 16 18 20 22 24
0 2 4 6 8 10 12 14 16 18 20 22 24 1
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17 19
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17 19 21
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17 19 21 23
Resize function works properly
Number 6 present in Array
Number 30 not in Array

Breakpoint 1, remove_elem (arr=0x7fffffffdf30, target=21845) at terrible_dynamic_size_array_unsorted.c:61
61 int remove_elem(struct int_array* arr, int target) {
(gdb) s
63     unsigned int i = 0;
(gdb) s
65     if ((arr->count == 0))return FALSE;
(gdb) print arr->count
$1 = 25
(gdb) s
70     if (i == arr->count)return FALSE;
(gdb) print i
$2 = 0
(gdb) print arr->count
$3 = 0
(gdb) |
```

After a few unsuccessful runs of code we notice something is wrong with the remove function. We don't seem to know what yet, so we add a break to the remove function and step through it. If we check the `arr->count` variable value at the beginning of the call, it should be 25. If we check it at the end of the call, it becomes 0. Count should not be changed in remove so we look for where it might be and find that the first if statement that checks if the array is empty is set to `=` instead of `==`. Change that.

```
59 }
60
61 int remove_elem(struct int_array* arr, int target) {
62
63     unsigned int i = 0;
64
65     if ((arr->count == 0))return FALSE;
66
67     while (i < arr->count && arr->data[i] != target) i++;
68
69
70     if (i == arr->count)return FALSE;
71
72     arr->data[i] = arr->data[arr->count];
73
74     arr->count--;
75
76     return TRUE;
77 }
78
```

Error 6:

```
Breakpoint 1, remove_elem (arr=0x7fffffffdf10, target=21845) at terrible_dynamic_size_array_unsorted.c:61
61 int remove_elem(struct int_array* arr, int target) {
(gdb) s
63 unsigned int i = 0;
(gdb) s
65 if ((arr->count == 0))return FALSE;
(gdb) s
72 return FALSE;
(gdb) s
```

Running “remove\_elem” again we still don’t get the correct result so we check for more errors. We find that for some reason remove is skipping over the “if(i == arr->count)” block of code on line 70 and notice it has ANOTHER hidden semicolon. Once we remove it, the code finally works as intended.

```
int remove_elem(struct int_array* arr, int target) {
    unsigned int i = 0;

    if ((arr->count == 0))return FALSE;

    while (i < arr->count && arr->data[i] != target) i++;

    if (i == arr->count);
    {
        return FALSE;
    }

    arr->data[i] = arr->data[arr->count];

    arr->count--;

    return TRUE;
}
```

**AND OUR CODE WORKS AS INTENDED YAY!!!**

```
Empty array created
Array cleared of data
0
0 2
0 2 4
0 2 4 6
0 2 4 6 8
0 2 4 6 8 10
0 2 4 6 8 10 12
0 2 4 6 8 10 12 14
0 2 4 6 8 10 12 14 16
0 2 4 6 8 10 12 14 16 18
0 2 4 6 8 10 12 14 16 18 20
0 2 4 6 8 10 12 14 16 18 20 22
0 2 4 6 8 10 12 14 16 18 20 22 24
0 2 4 6 8 10 12 14 16 18 20 22 24 1
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17 19
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17 19 21
0 2 4 6 8 10 12 14 16 18 20 22 24 1 3 5 7 9 11 13 15 17 19 21 23
Resize function works properly
Number 6 present in Array
Number 30 not in Array
Number 23 removed from Array
Number 24 removed from Array
Number 0 removed from Array
Number not in Array
Array destroyed
user@box:~/c/Lab4$
```