Vraj Patel
400434795
patev61

Debugging Report

Syntax Errors:

Identification of Bug:

This bug was solved through inspection, a missing semicolon will not allow the file to correctly compile, therefore running into an error:

```
Question4.c: In function 'sortDatabyBubble':
Question4.c:45:17: error: expected ';' before 'array'
                  array[i + 1].charData = array[i].charData;
```

Proposal of Bug Fix:

Simply just add a semicolon to fix the syntax error:

```
array[i + 1].intData = array[i].intData; //added semicolon for compilation
array[i + 1].charData = array[i].charData;
```

Validation of Fix:

The code now compiles without any errors from the compiler, allowing access to run the file and continue debugging in gdb.

Semantic Errors:

Bug 1

Identification of Bug:

During the 3rd iteration through the loop, intData was constantly being overwritten by the same value each time (the starting value of 10)

```
curr = 77
next = 6
done = 0
(gdb) p *array@size
$3 = {{intData = 10, charData = 99 'c'}, {intData = 10, charData = 99 'c'}, {intData = 10, charData = 99 'c'}, {intData = 12,
    charData = 122 'z'}, {intData = 77, charData = 97 'a'}, {intData = 77, charData = 97 'a'}}
(gdb)
```

In the code given, array[i] is set to temp, but temp isn't being changed since at the top of the loop, temp is being initiated to the same value, and that is why it is being overwritten:

Vraj Patel
400434795
patev61

```
    if(curr > next)
    {
        temp.intData = array[i].intData;
        temp.charData = array[i].charData;

        array[i].intData = temp.intData;
        array[i].charData = temp.charData;

        array[i + 1].intData = array[i].intData; //added semicolon for compilation
        array[i + 1].charData = array[i].charData;

        done = 0;
    }
```

This shows that there is no swapping done and only overwriting.

Proposal of Bug Fix:

The initial way was thinking of correct, but instead of constantly storing array[i] as its temp value, we should set it to the next value of the array which is [i+1]. Consequently, array[i] should be set with the next element array (array[i+1])

```
        // Swap two elements whenever current element value is larger than the value
        if(curr > next)
        {
            temp.intData = array[i].intData;
            temp.charData = array[i].charData;

            array[i].intData = array[i+1].intData;
            array[i].charData = array[i+1].charData;

            array[i + 1].intData = temp.intData; //added semicolon for compilation
            array[i + 1].charData = temp.charData;

            done = 0;
        }
    }
```

Vraj Patel
400434795
patev61

Validation of Bug Fix:

The output matches the array within the testcases.c file (except there is one more added element):

```
(gdb) p *array@size
$10 = {{intData = -42, charData = 63 '?'}, {intData = -5, charData = 107 'k'}, {intData = 2, charData = 66 'B'}, {intData = 6,
    charData = 8 '\b'}, {intData = 10, charData = 99 'c'}, {intData = 12, charData = 122 'z'}}
(gdb) 1
```

```
struct Q4Struct expected[]={{-42, '?'}, {-5, 'k'}, {2, 'B'}, {10, 'c'}, {12, 'z'}, {77, 'a'}};
```

Despite of the extra added index, the array is still perfectly in ascending order as expected to be.

Bug 2:

Identification of Bug:

Compared to the expected result, there is an extra index added (intData = 6). This extra index is a "garbage value" meaning that there is some sort of over indexing involved.

```
(gdb) p *array@size
$10 = {{intData = -42, charData = 63 '?'}, {intData = -5, charData = 107 'k'}, {intData = 2, charData = 66 'B'}, {intData = 6,
    charData = 8 '\b'}, {intData = 10, charData = 99 'c'}, {intData = 12, charData = 122 'z'}}
(gdb) 1
```

```
struct Q4Struct expected[]={{-42, '?'}, {-5, 'k'}, {2, 'B'}, {10, 'c'}, {12, 'z'}, {77, 'a'}};
```

```
.....................FF..

There were 2 failures:
1) TestQ4_BubbleSort_1: testCases.c:319: expected <0> but was <72>
2) TestQ4_BubbleSort_2: testCases.c:335: expected <8> but was <11>

!!!FAILURES!!!
Runs: 26 Passes: 24 Fails: 2
```

This following screenshot shows the identification of the "garbage value" (it will be 6). Please note that I = 5 is my last index and the next value is 6 which is a garbage value:

```
(gdb) i locals
temp = {intData = 77, charData = 97 'a'}
i = 5
curr = 77
next = 6
done = 0
(gdb)
```

Vraj Patel
400434795
patev61

Proposal of Bug Fix:

Since the function is going through an extra index, we need to “take an index away” from the function so it doesn’t produce garbage values. This is easily solvable by doing the following:

```
for(i = 0; i < size - 1; i++)
{
    curr = array[i].intData;
    next = array[i + 1].intData;
```

In the following picture, the equal sign was removed which “takes away an index.”

Validation of Bug Fix:

```
temp = {intData = 10, charData = 99 'c'}
i = 4
curr = 12
next = 77
done = 0
(gdb)
```

This screenshot validates that the bug is fixed since the i = 4 (the maximum is goes to) and the next value isn’t a garbage value anymore and it is a value which exists within my array.

```
PS C:\Users\vrajp\OneDrive\Desktop\2nd year\1st Sem\2SH4 - Intro to CompEng\Lab2\lab2-patev61> ./Lab2
.........................

OK (29 tests)
```

The code also solves all the test cases including the new ones including by me.