Sebastian & Fabian
Lab 9 - CIS 452

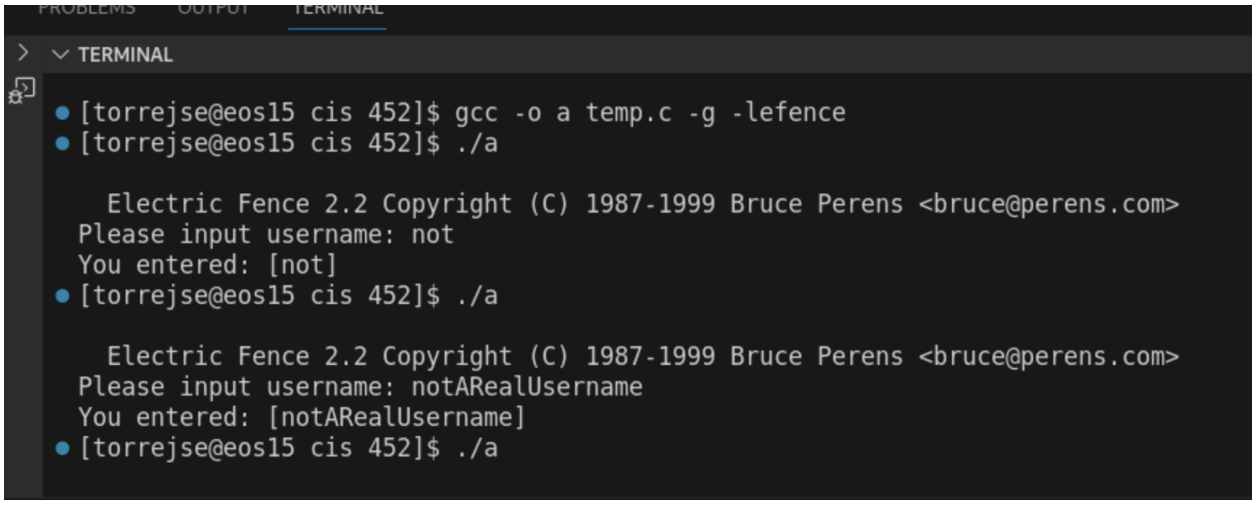
1. Describe the error precisely (nature of problem, location). Fix the problem and submit your corrected source code along with a screenshot of the execution. Develop a robust solution that allows the user to enter any size username (within reason) without massively over allocating memory. (10 points)

```
kirbergf@arch02:~/cis452/lab09$ vim sampleProgramOne.c
kirbergf@arch02:~/cis452/lab09$ gcc sampleProgramOne.c -o sampleProgramOne
kirbergf@arch02:~/cis452/lab09$ ./sampleProgramOne
Please input username: username
You entered: [username}
kirbergf@arch02:~/cis452/lab09$ ./sampleProgramOne
Please input username: notARealUsername
You entered: [notARealUsername}
kirbergf@arch02:~/cis452/lab09$ gcc sampleProgramOne.c -o sampleProgramOne -g -l
efence
kirbergf@arch02:~/cis452/lab09$ ./sampleProgramOne

  Electric Fence 2.2 Copyright (C) 1987-1999 Bruce Perens <bruce@perens.com>
Please input username: username
You entered: [username}
kirbergf@arch02:~/cis452/lab09$ ./sampleProgramOne

  Electric Fence 2.2 Copyright (C) 1987-1999 Bruce Perens <bruce@perens.com>
Please input username: notARealUsername
Segmentation fault (core dumped)
kirbergf@arch02:~/cis452/lab09$
```

The error in the program is that the memory malloced to the data1 variable is set to be of 16 bytes due to the SIZE definition. When attempting to enter a username that exceeds this size, such as in the case of entering the username notARealUsername, while the program is compiled with the Electric Fence library, the program encounters a segmentation fault error.



2. What's going on here, and why? (5 points) Hint: google "address obfuscation"

The base addresses of data segments are being changed randomly. The primary purpose of this is data security. Certain software attacks may attempt to exploit memory programming errors, such as buffer overflows, in an attempt to gain sensitive information or knowledge regarding the function of the program. This process is much easier to perform if the attacker has a good understanding of the details and memory locations of their targeted programs. By randomizing memory addresses, it is more difficult for malicious actors to gain access to such resources.

3. Submit the source code you used along with the memory map diagram mapping out all

of the aforementioned regions. Submit a screenshot of your program execution. (45 points)

Stack - Address: 0x16b61ef98, Depth: 5 **Stack - Address: 0x16b61ef68, Depth: 4** **Stack - Address: 0x16b61ef38, Depth: 3** **Stack - Address: 0x16b61ef08, Depth: 2** **Stack - Address: 0x16b61eed8, Depth: 1** **Stack - Address: 0x16b61eea8, Depth: 0** **Heap - Address: 0x131e060a0, Depth: 0** **Heap - Address: 0x131e06090, Depth: 1** **Heap - Address: 0x131e06080, Depth: 2** **Heap - Address: 0x131e05fb0, Depth: 3** **Heap - Address: 0x131e05fa0, Depth: 4** **Heap - Address: 0x131e05ef0, Depth: 5**	**STACK (From top to bottom)** **HEAP (From bottom to top)**
globalUninitializedVariable2: 0x1047e800c **globalUninitializedVariable: 0x1047e8008** **globalInitializedVariable2: 0x1047e8004** **globalInitializedVariable: 0x1047e8000**	**DATA (Initialized variables first, and then unititialized variables)**
0x10128f001	**TEXT**
0x1000004d3	**LIBRARIES**

```
seabass@Sebastians-MacBook-Air Lab 9 % ./a
Addresses:
globalInitializedVariable: 0x1047e8000
globalUninitializedVariable: 0x1047e8008
globalInitializedVariable2: 0x1047e8004
globalUninitializedVariable2: 0x1047e800c
Stack - Address: 0x16b61eea8, Depth: 0
Stack - Address: 0x16b61eed8, Depth: 1
Stack - Address: 0x16b61ef08, Depth: 2
Stack - Address: 0x16b61ef38, Depth: 3
Stack - Address: 0x16b61ef68, Depth: 4
Stack - Address: 0x16b61ef98, Depth: 5
Heap - Address: 0x131e060a0, Depth: 0
Heap - Address: 0x131e06090, Depth: 1
Heap - Address: 0x131e06080, Depth: 2
Heap - Address: 0x131e05fb0, Depth: 3
Heap - Address: 0x131e05fa0, Depth: 4
Heap - Address: 0x131e05ef0, Depth: 5
seabass@Sebastians-MacBook-Air Lab 9 %
```