Etude 13 – Bug Squashing
Trent Lim 5884658

Dear Junior Developer,

After reviewing your code, you have made a good effort on the program, but there are a few areas of improvement which need to be addressed. These areas of improvement include both stylistic and syntax errors, many of which made your code difficult to understand:

- Lack of commenting
- Lack of meaningful variable/function names
- Syntax errors
- Stylistic issues
- Efficiency
- Memory leakage
- Sanity Checking
- User experience

Below is a further breakdown of these issues.

Lack of Commenting
I noticed that you did not have a single comment in your code. Comments are helpful, not just for others reading your code, but for you to understand it too. Comments should help clarify the intent of your code and improve understanding. It was difficult for me to understand your code without comments, and I needed to read every line of code to decipher what was happening and resolve any issues.

Lack of Meaningful Method Names
All of your method names need to be more meaningful to help other developers understand your code. Again, I had to read every line of code in your methods to understand what each function is doing. In the future, please name your methods so that they clearly state what the method is doing. For example, I changed the name of one of the functions from `ffn to find_first_name` which is less ambiguous.

Syntax Errors
Your code had many syntax issues which prevented compilation. In the future, I suggest you change the way you compile your code in the makefile so that any errors/warnings when compiling will be shown to you. Adding the `-Wall` argument will do this for you. This would have shown you multiple errors in your code that were easily resolved.

Stylistic Issues
There was a spelling mistake in your code which would not affect the functionality of the code and is a small issue, but it is good practice to ensure that words are spelled correctly.

Additionally, correct formatting of your code would help me understand it easier. For example, in your switch statement, I would appreciate it if the code for each case was indented, as this would improve legibility.

Efficiency
Every time your program searched for a person, it would sort the array. This is an inefficient use of resources, as sorting an array: 1. has a large time complexity and 2. is completely unnecessary when searching for an instance in a list in this case.

Memory Leakage
Your program had a few issues concerning the leakage of memory. When you allocated memory for the `val` variable, the memory was allocated within the loop, meaning that a new chunk of memory was allocated every time the loop was run. This means that the memory allocated in previous loops is inaccessible as `val` points to a different memory block, and is completely unused after the loop that it is allocated in. Clearly, this wastes a lot of memory on the computer running the program as much of the memory allocated is useless.

Since `val` is a variable that only stores one value, memory for the variable should be allocated outside the loop, and every time the variable changes, it uses the same chunk of memory.

Sanity Checking
With the code that you sent me for review, any user that uses your program can enter literally anything as an input and may exploit this security issue for malicious purposes. You should put restrictions on what the user can input and inform them if their input is/is not acceptable.

For example, your code has a buffer of size 10 in which the user's query will be stored. However, you use the `gets()` function which does not allow you to specify the maximum number of characters allowed in the input. I suggest you replace this with `fgets()` – or `scanf()` if you only want to get input from `stdin` - and specify 10 as the maximum number of characters to restrict what users can input into the program.

User Experience
After I fixed these issues, your program's user interface did not make it for a user to understand how to use it. When your program requires user input, your program should tell the user what input is required. For example, when the user needs to input the command stating which type of user information to search by, they should be informed of what each command does.

Furthermore, the purpose of the program is to find the user by a specific type of information, e.g., their first name. When the program finds this user, it should print out their information instead of saying whether the user exists or not. This is not a big problem, but it worsens the user experience of your program.

List of changes made
- Replaced all instances of `emialAddress` with `emailAddress`
- Add comments to code
- Changed method names
- Removed sort methods
- Added {} braces to for/if statements
- Set i to -1 every time a find method is called
- Used `strcmp` to see if two strings are equal instead of ==
- Fixed methods returning 1 instead of i
- Changed methods to return -1 instead of 0 if the person is not found
- Fixed indentations
- Fixed initializing `count` twice
- Close file after reading input to free memory
- Added an if statement to inform the user if the input file is not found
- Move memory allocation for `val` outside of loop

- Move `command` declaration outside of loop
- Add print statements to inform user of input requirements
- Add print statement to tell user that the program is exiting
- Add code for program to print the details of the person that the user is querying

Conclusion
Your program now works without issues and has a more informative user interface. I hope that you have learned from my suggestions and will avoid these issues in the future. I stress that you learn more about memory allocation, literate programming, and how to compile your program correctly so that any issues with your code are brought up. Hope this helps!

Kind regards,
Trent