# COSC326 Etude 13:

Dear Bartholomew.

We have completed a code review for you. A number of general comments stand out from the overall picture described in the body paragraph and I've listed them below in greater detail.

The code was incomplete and inconsistent but a good start for an implementation of a database with query capabilities. Segmentation faults and no comments / information in method names make it hard to understand the purpose of the code but after a bit of work I could see the foundations you have made for the database implementation. The code is lacking comments and you have a method that does the same thing for four different variables (first name, last name, email address and phone) which can be changed into one method that reads in which variable you want to sort. Not freeing memory is not good coding practice since it will lead to leaks that make your code vulnerable and can also slow the user's computer. There is no input validation so users have free reign over what they put into the code, and if an error occurs, it will not tell them any error messages, so they won't know what they have done or how to fix it. Documentation is always good to add with a readme explaining the task and goal of the code project so people would understand the task, Testing is also a great thing to do when starting a new method or block of code you feel might throw an error, doing error checking ensures when you run the code, you have an isolated part to error check instead of the whole code. If you have any more questions, feel free to come to me and I can walk you through my process of eliminating the bugs and getting the code working.

Thanks, Aidan.

## Removing unnecessary variables

- I do not see the point in these variables since the only use was in the for loops, which we can declare these variables in the loops anyways.
- Count is not used in the code apart from the for loop that increments after and some redundant methods.
- This makes the code cluttered and longer than it has to be which ruins the clarity of the code.

## Naming conventions

- These names were not informative at all and have all been changed to better describe their functions
  - Sfn = sort_first_name
  - Ffn = find_first_name
  - Sln = sort_last_name
  - Fln = find_last_name
  - Sem = sort_email_address

  - Fem = find_email_address
  - Sph = sort_phone_number
  - Fph = find_phone_number

## New emalloc function / allocating memory problems

- This will better allocate memory for variables and will return an error if we are allocating to a NULL variable. This ensures the code will only occupy a necessary amount of memory and will also not give memory if values are NULL.
- In the for loop which assigns data to s, then assigns s to ss, I can see that you have forgotten to allocate memory for phone so to fix that we just use our emalloc function to allocate it memory.
- I've removed the multiplication on most emalloc variables. This is because I feel it uses a lot of extra memory that isn't needed to use and can potentially clog the system.
- Freeing variables mean we don't have any memory leaks so variables that have used either malloc or the emalloc function written in this code need to be freed e.g., s->firstname, s->lastname, s->emailAddress, s->phone, s and val
- Val was an issue inside the while loop since it was being reallocated every time we went through the loop causing major leakage and becoming a serious problem. To fix this, I moved it to the top of the main where you should always declare variables to make the code clear and concise

## Implementing new variables / methods

- Adding linecount ensures that we are only adding data to ss and not any blank entries like we had before
- The old sort function was too hard to understand and was different for every sort method so I wrote a new one. I've also changed count to linecount.
- Adding the if statement to check the args will mean if the user doesn't enter a test file, they will get a message to tell them how to use the program instead of an error.
- Adding a way to exit the code means the user can enter '0' to quit out of the program instead of using a kill command. This is good practice so people with no knowledge of programming have a way to stop the program
- This is good coding practice to be consistent with variables.

## Major changes to code

- S** was too complex and clunky to understand and use and from what from what I've seen they perform the same task of holding instances of s
- Changing the for loop for reading in data to the dynamic variable linecount is much smarter because it means we are only storing the data we need and no extra space is wasted.
- Increasing the buffer from 10 to 50 ensures we don't have an overflow error and since an email is usually more than 10 characters, an error would always occur.
- Removing the multiplication of emalloc variables is because it uses a lot of extra memory that isn't needed to use and can potentially clog the system.
- Removing strcpy where it isn't needed is because when we fill the buffer, we fill it with the response of what switch case to go into, not the value we are searching for. To fix this, I've added another fgets in the switch cases asking

the user to enter the data they wish to search for and removed redundant ones.

- Reworking the switch cases to give the user better information means the user understands what is going on and is not confused at any stage. With each switch case now prompting the user to enter data, coping the data from the buffer to val, sorting the data properly and outputs whether the data is found or not, they know what is happening at every stage because of the print statements informing them.
- When the user quits, they now get a goodbye message, and the program will return EXIT_SUCCESS which is good coding practice.
- The old make file used math which wasn't needed and also had 2> /dev/null making it so we couldn't see the errors of the program which makes it harder to diagnose problems.
- The old find methods used "==" to compare the value we are looking for e.g., ss[i]->firstname with val. This is bad coding practice since we have a function (strcmp) to compare strings more efficiently.
- Spelling mistakes are a small problem but once again, bad coding practice since having variables with meaningless / incorrect names makes it harder to read the code for someone else who might need to alter / fix mistakes in it later.

## Lack of comments

- Comments are an essential tool for a developer because it's an easy way to explain to another developer, the meaning and purpose of methods and blocks of code.
- Since the code had zero comments, I couldn't understand what the I / O should be for the code or even what the code was going to do. This is very bad coding practice and needs to be addressed immediately.

## User interface issues

- The user interface was in shambles with no information at any step in the program. I've added a menu telling the user the commands and their purpose, an exit command so the user doesn't get stuck in the code, information about when to enter input.

## Sanity of user input checks

- User input checks are essential for effective code, since the user may accidently enter in wrong input or not understand the format we are looking for. You should always inform the user e.g., when they are meant to run the code with a test file.
- Using fgets instead of gets is essential and a major problem because (gets) has no way to stop buffer overflow error so we use fgets to ensure this doesn't happen
- Fscanf also has the problem of not checking user input so the user could have a file that isn't formatted correctly, and the code won't tell the user which means when the output is given the user, they won't understand what they did wrong