COMP10050
Assignment 1 – Code Design Decisions
Evan Spendlove (18492656)

General

1. **Why did I use structures for this assignment?**
    a. I created a type-defined structure called '*fb_user*' in my assignment. I used *typedef* because it shortens the code I have to write every time I use the structure fb_user. Instead of having to write *struct fb_user*, I can simply write *fb_user*.
    b. I used a structure because it allowed me to store a user as a single variable, and to include all relevant information as attributes of that variable. I included the username, list of friends and friend count of each user as their attributes. This made moving users around much simpler. It also meant I didn't have to swap all of the elements associated with a user when I wanted to swap two users in an array. I could simply swap one entire structure with another. This made coding the rest of the assignment much easier.
2. **Why did I define my constants using *static const*?**
    a. Why define constants?
        i. By defining constants for variables such as max number of users, max number of friends per user, max username character count, etc. I was able to make the program more flexible. If you want to change the maximum number of users permitted, you simply have to change the *static const int* defined in the userStructure header file. This makes adapting the program to the ever-changing needs of the client much easier.
    b. Why *static const* over #define?
        i. I chose not to use *#define* because it a pre-processor directive, which means that the compiler completes this instruction before actually compiling the rest of the file. This can cause problems with local versus global scope variables, since a *#define* constant will always override a local variable. Also, it is much harder to debug a program using *#define*.
        ii. I chose to use *static const* instead, because it allows me to override this global variable with local variables where necessary. Declaring the variable as static means that it is only defined once during the program, which is excellent, since I use these variables multiple times throughout my program. Declaring the variable as a constant means that the compiler will throw up an error if I try to modify the variable, which is very convenient for debugging the program.
    c. Why did I choose to place my constants in the userStructure header file?
        i. I chose this location as this header is included in all of my other files, so it ensures I won't be redefining any of these constants in any other location. Also, grouping the definition of constants makes it easier to change them all at once if you need to.

COMP10050
Assignment 1 – Code Design Decisions
Evan Spendlove (18492656)

Input

3. **Why did I use *fgets()* instead of *scanf()*?**
    a. I used *fgets()* because it will read in input up to a pre-set character limit or until a newline character is entered. This is beneficial, because I don't have to use loops with *scanf()*, which can be messy and can provide undesired behaviour. When checking if a newline character is stored, I can simply check the first character of the previous entry and see if it a newline. Also, keeping the newline character at the end of each username and friend name makes printing easier, in my opinion.
4. **Why did I use *strcpy()* over *for()* loops?**
    a. *Strcpy()* is, in my opinion, a far cleaner and more logical way to copy strings from one variable to another. As such, I avoided using *for()* loops to copy each individual character, and instead went for the *strcpy()* function, which made the process much easier.

Sorting

5. **Why did I use the quicksort algorithm?**
    a. One of the sorting libraries that I provided utilises the quicksort algorithm. I chose this sorting algorithm as it has an average case time complexity of O(n log (n)), which is better than both bubble sort and insertion sort, which both have an average case time complexity of O(n^2). Quicksort is consistent, in that the best and average cases are the same.
6. **Why did I use *strcasecmp()* over *strcmp()* for sorting alphabetically?**
    a. I used *strcasecmp()* because it ignores case when comparing, which allowed me to sort alphabetically. Initially, I looped over each username and made it all lowercase, then made the first letter of each word in the name uppercase. This can be avoided using *strcasecmp(),* hence why I went with that option.
7. **Why did I use sorting networks?**
    a. I provided a second library for sorting functions, which is more efficient in both space and time, but only works for a max of six users. I found sorting networks while researching efficient ways of sorting small data sets. I decided to build an alternative library that one could use if they kept within the limit of at most 6 users. While the library isn't entirely necessary, it was an interesting learning exercise and provides a more efficient sorting method for the specifications of this assignment. I went with it to maximise the efficiency of my program, and also to learn about the topic.
    b. By providing two libraries, I have a very efficient library for the program specifications provided, and a more flexible library, which can be utilised if the specifications change.