LAB01 Report - Debugging

Name: Johnson Umeike

Student ID: 3094849

Section One: Memory Leaks

1. uninitialized_variable was not initialized before it's use in the for loop on line 20. I initialized it within the **for** loop to 0.
2. definitely lost memory leak occurred because the dynamic memory (8 bytes for each block) used by double pointer ****definitely_lost** wasn't deallocated. I used **free(*definitely_lost)** to do this just before the end of the loop.
3. indirectly lost memory leak occurred because the dynamic memory (7 bytes for each block) used by pointer ***definitely_lost** wasn't deallocated. I used **free(definitely_lost)** to do this just before the end of the loop.
4. still reachable memory leak was resolved by freeing the 42 bytes dynamic memory allocation specified by **still_reachable = malloc(42)**.
5. possibly lost memory leak occurred because an offset of 4byte was added to the void pointer possibly_lost. Because you can't deallocate memory using a pointer pointing to the middle block, I used **possibly_lost -= 4** to take the pointer back to the start of the block on the memory heap and finally used **free(possibly_lost)** to deallocate the initial memory allocation.

Section Two: Bug Fixes

1. **echo()** routine needed a NULL character to exit the recursion but didn't get it because the **bug_info.sentence** array is not null-terminated. This was resolved by increasing the length of the sentence array size from 6 to 7. Next, we add **bug_info.sentence[7] = NULL** on line 107.
2. Garbage was being displayed on the screen because **echoohce()** routine had a bug. The format type "%s" (string) is not the same as the variable iter (array of strings). I changed iter to *iter on line 80 to resolve this.
3. **echooehce()** routine printed scary bug ("~~~~~~~~ SPIDER!!! ~~~~~~~~") because of an overflow. The loop condition required a NULL character at the beginning of the array to exit. However, this is not applicable. Hence, I used the predefined variable **stop_beginning** which is (strs – 1) or the pointer to the first byte of memory address before strs.
4. **bug_info.num_bugs_on_mars** wasn't initialized at the correct location which made its use in line 111 to produce a wrong result. I had to initialize it earlier under initialize structure on line 91. This corrected the error.
5. I changed bug_info.sentence[2] to bug_info.sentence[6] to print desired output of (null).

Section Two: Memory fix for bug.c

1. There was a direct memory leak because dynamic memory was not deallocated before another allocation by **bug_info.sentence[2] = strdup("colorful").** I used free(bug_info.sentence[2]) on line 130 first to deallocate the space initially allocated by bug_info.sentence[2] = strdup("useless") on line 102.
2. use of **free()** routine in **free(bug_info.useless_bug)** isn't required and caused a program crash by truncating the output of the last printf() on line 159.