Name: Henri Malahieude (hmala001) SID: 862268736

1. Run Prog-1
    a. Line 11 of Prog-1.cpp has a segmentation fault
    b. It seems variable i of the loop has value 18446744073709541929, which may be due to an integer rollover issue. We would need to change the direction that the for loop is going through, or to change size_t to an typethat can handle negatives.
2. Run Prog-2
    a. According to valgrind we get a “stack smashing” and it terminates the program with SIBABRT. This may be due to improperly sized arrays, as well as trying to push a char pointer through cout.
    b. Valgrind reports line 18 as having the issue, which is a function that has hard-coded array sizes of 8. If any string passed has a size larger than 8, then it will have a memory issue. Therefore we can have the arrays scale dynamically with the size of the string, and use a for loop to output each character.
3. Run Prog-3
    a. It never ends, and I had to terminate it with SIGINT. It most likely would’ve failed in a Stack Overflow or something related to no longer having any usable memory in the system. Another possible error is SEGFAULT due to the resize potentially accessing further than the original array allowed.
    b. It seems to be that the resize function is only 1 by 1, and then copying each data element sequentially. This is basically a factorial run time O(n!). Instead, it should multiply the wanted space by 2 first instead of just incrementing by 1.
4. Run Prog-4
    a. The program is SEGFAULT-ing at line 56, where there is “n->next->prev” implying that either next or prev does not exist.
    b. The program crashes in two cases, *add_after* doesn’t check that it’s adding after a tail, *remove* isn’t properly formatting in the edge cases of tail or head. Add_after, and Remove were given slight changes for the edge cases.
    c. The destructor wasn’t working properly and deleting unnecessary information. It was rewritten to only delete information held within the linked list.
    d. Valgrind didn’t display an error in my case, but a potential one could’ve been trailing pointers because of the remove function. However, that was dealt with earlier in the edge case handling.
5. Run prog-5
    a. First issue is in *array(size_t size)* where the data constructor uses an uninitialized value.
    b. Second issue is that an *operator=()* function wasn’t overloaded, so it simply SEGFAULT-ed due to no space in a 0 size array.
    c. Third issue is a memory leak which means we aren’t deleting information somewhere;
6. Run prog-6
    a. We get an invalid write of size 2. Where we set the name of an object. It is explicitly where the name of the object is set. The issue seems to be about trying to access an address in the assembly instruction *“mov %ecx,(%rdi)”* at line 341 of

__memmove-vec-unaligned-erm__.s while rdi is 0x00. Causing a SEGFAULT since 0x00 is the null pointer in C.

b. To stop this issue, unfortunately I don't know what exactly goes on. I just know that there seems to be an unaligned memory space that can't be fixed while creating a new string. Possibly because of the space given to the string with the sphere.