With GDB

1. Fix all the “.” to “🡪” for line 26, 27, and 35. As it points to the object inside the node class. Include the “vector” library on the top. Change the node class from private to struct(public class).
2. Using GDB and find out there is segmentation fault in first for loop of create\_LL function. The mylist vector de-reference as a value instead of as a pointer. To fix that, initialize all the node in mylist vector as “new node()”.
3. Using GDB again and find out there is segmentation fault in the while loop of the sum\_LL function, the node out of the bound before iterate through every value in it. According to the hint, there was an error occur in the second part of function create\_LL. The error is that the loop will go out of the bound before it reaches to the last index, to fix that I just change “i < node\_num” to “i < node\_num-1”.
4. Lastly, Using GDB again and find out LeakSanitizer has encountered a fatal error.To avoid any potential memory leak, I write a for loop to free all the allocated memory that was use in the previous function.

With AddressSanitizer

1. Fix all the “.” to “🡪” for line 26, 27, and 35. As it points to the object inside the node class. Include the “vector” library on the top. Change the node class from private to struct(public class).
2. Using AddressSanitizer and find out the program was abort by the segmentation fault. There is one error occur in the first for loop of the create\_LL function. Another error occur in the main function where the create\_LL function was called. The mylist vector de-reference as a value instead of as a pointer. To fix that, initialize all the node in mylist vector as “new node()”.
3. Using AddressSanitizer again and find out the program was abort heap-buffer-overflow on address 0x603000000028. There is one error occur in the second for loop of the create\_LL function. Another error occur in the main function where the create\_LL function was called. The error was caused by the loop will go out of the bound before it reaches to the last index, to fix that I just change “i < node\_num” to “i < node\_num-1”.
4. Lastly, use the AddressSanitizer again and find out the program was abort LeakSanitizer: detected memory leaks. To avoid any potential memory leak, I write a for loop to free all the allocated memory that was use in the previous function.