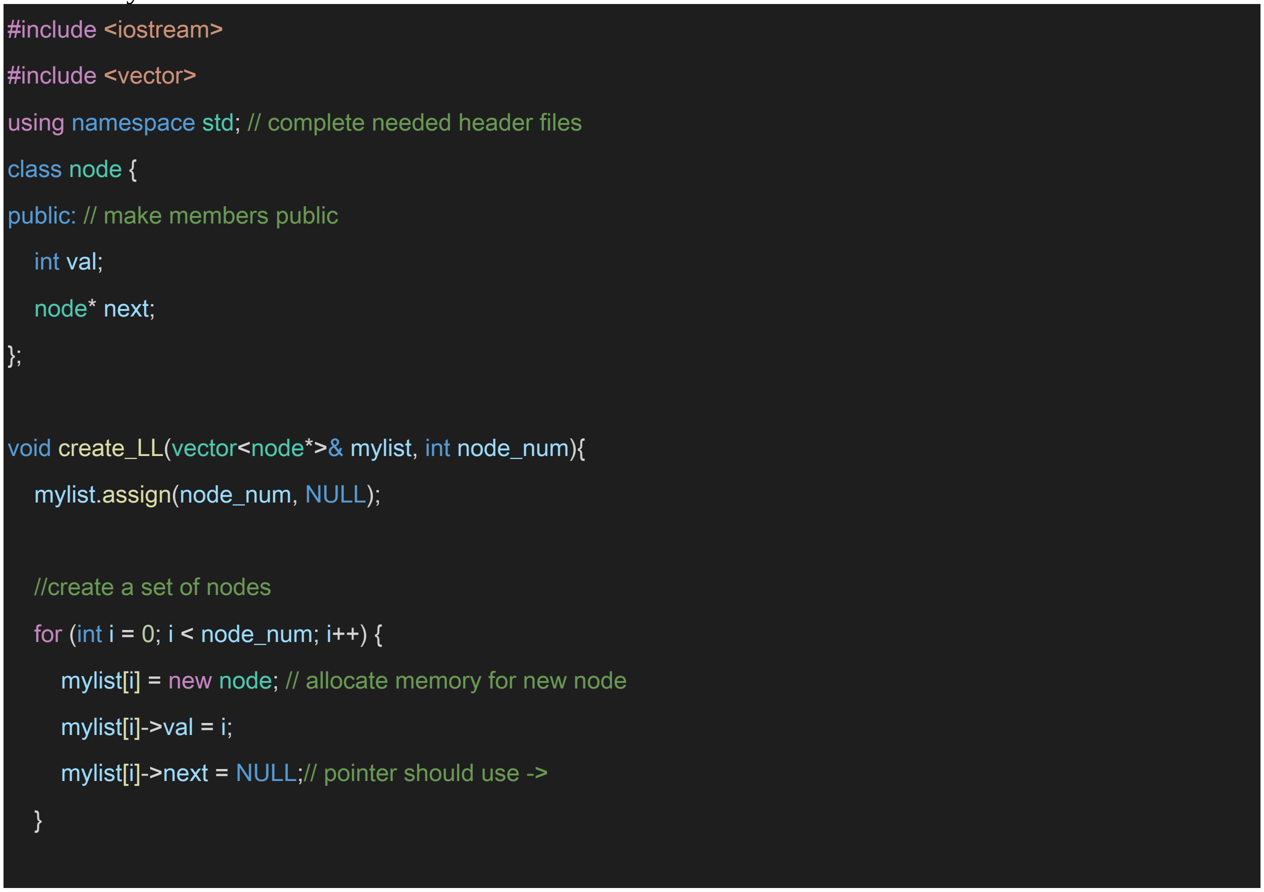
Name: Arthu Chen
UIN: 327003368

Host: Mac
Guest: Linux

These are the steps I took:

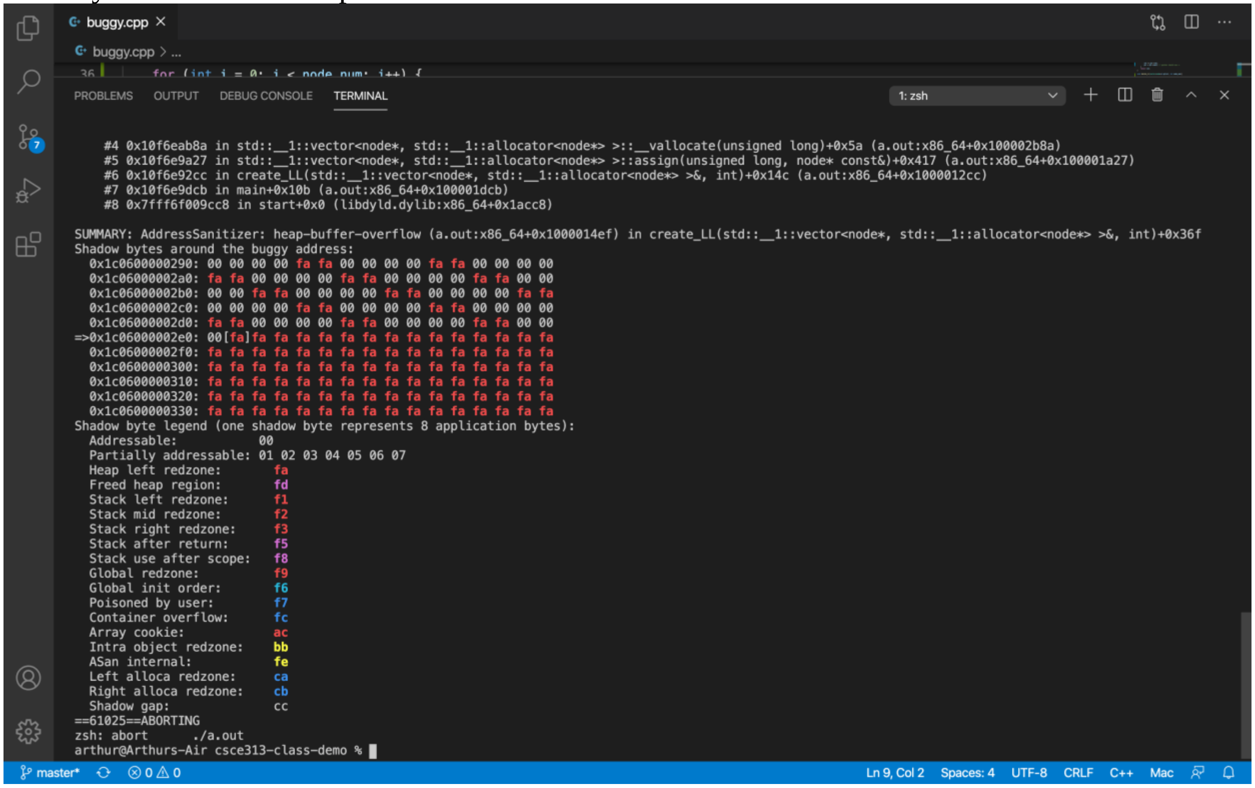
1. Download osboxes then ubuntu for linux OS.
2. Enable shared folder between host and guest
3. Download g++, gdb, etc
4. In guest terminal, following the video given by professor and with the help from gdb:
    a. Try to compile buggy.cpp after learning multiple commands in gdb, such as break for breakpoint, print to print variable value, and -g for symbol table
    b. found seg fault occurs around line 16
    c. Then fix part C by allocating space for each node
    d. Since mylist[i] are pointers, I replaced "." with "->".
    e. Wrote a function to delete all the notes to avoid memory leak.
5. As I run g++ buggy.cpp -fsanitize=address, the Asan says the error was caused by memory access at which function thus its memory address.

This is my corrected code:



```
#include <iostream>
#include <vector>
using namespace std; // complete needed header files
class node {
public: // make members public
    int val;
    node* next;
};

void create_LL(vector<node*>& mylist, int node_num){
    mylist.assign(node_num, NULL);

    //create a set of nodes
    for (int i = 0; i < node_num; i++) {
        mylist[i] = new node; // allocate memory for new node
        mylist[i]->val = i;
        mylist[i]->next = NULL;// pointer should use ->
    }
    //create a linked list
    for (int i = 0; i < node_num; i++) {
        mylist[i]->next = mylist[i+1];// pointer should use ->
    }
}

int sum_LL(node* ptr) {
    int ret = 0;
    while(ptr) {
        ret += ptr->val;
        ptr = ptr->next; // pointer should use ->
    }
    return ret;
}

void delete_LL(vector<node*>& mylist, int node_num){
    for (int i = 0; i < node_num; i++) {
        delete mylist[i];
    }
}

int main(int argc, char ** argv){
    const int NODE_NUM = 3;
    vector<node*> mylist;

    create_LL(mylist, NODE_NUM);
    int ret = sum_LL(mylist[0]);
    cout << "The sum of nodes in LL is " << ret << endl;

    //Step4: delete nodes
    delete_LL(mylist, NODE_NUM);// to delete all nodes
}
```

And my screenshot for step 10:



```
    for (int i = 0; i < node_num; i++) {
```

```
    #4 0x10f6eab8a in std::__1::vector<node*, std::__1::allocator<node*> >::__vallocate(unsigned long)+0x5a (a.out:x86_64+0x100002b8a)
    #5 0x10f6e9a27 in std::__1::vector<node*, std::__1::allocator<node*> >::assign(unsigned long, node* const&)+0x417 (a.out:x86_64+0x100001a27)
    #6 0x10f6e92cc in create_LL(std::__1::vector<node*, std::__1::allocator<node*> >&, int)+0x14c (a.out:x86_64+0x1000012cc)
    #7 0x10f6e9dcb in main+0x10b (a.out:x86_64+0x100001dcb)
    #8 0x7fff6f009cc8 in start+0x0 (libdyld.dylib:x86_64+0x1acc8)

SUMMARY: AddressSanitizer: heap-buffer-overflow (a.out:x86_64+0x1000014ef) in create_LL(std::__1::vector<node*, std::__1::allocator<node*> >&, int)+0x36f
Shadow bytes around the buggy address:
  0x1c0600000290: 00 00 00 00 fa fa 00 00 00 00 fa fa 00 00 00 00
  0x1c06000002a0: fa fa 00 00 00 00 fa fa 00 00 00 00 fa fa 00 00
  0x1c06000002b0: 00 00 fa fa 00 00 00 00 fa fa 00 00 00 00 fa fa
  0x1c06000002c0: 00 00 00 00 fa fa 00 00 00 00 fa fa 00 00 00 00
  0x1c06000002d0: fa fa 00 00 00 00 fa fa 00 00 00 00 fa fa 00 00
=>0x1c06000002e0: 00[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x1c06000002f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x1c0600000300: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x1c0600000310: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x1c0600000320: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x1c0600000330: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==61025==ABORTING
zsh: abort      ./a.out
arthur@Arthurs-Air csce313-class-demo % 
```