Andrew Lau

Exercise 1:

1. To fix the error we changed the values of V0 to 3, V1 to 3, V2 to 1 and V3 to 3:

Code:

#define V0 3

#define V1 3

#define V2 1

#define V3 3

1. The only distinct value needed for the preprocessor macro in this code is #include.
2. The -o flag compiles the file and creates an object file. When you make changes to the code, and rerun with the –o flag, the new code will compile.

Exercise 2:

1.

To set the breakpoint at main, type ‘break main’ and then run the code.

2. gdb commands

1. gdb  --args (filename) arguments

2. break if (statement)

3. next

4. step

5. continue

6. print (expression)

7. display

8. info locals

9. quit

Exercise 3:

The bug in the code was a logic error. The original code only checked if “a” was null, but did not check if “b” was null.

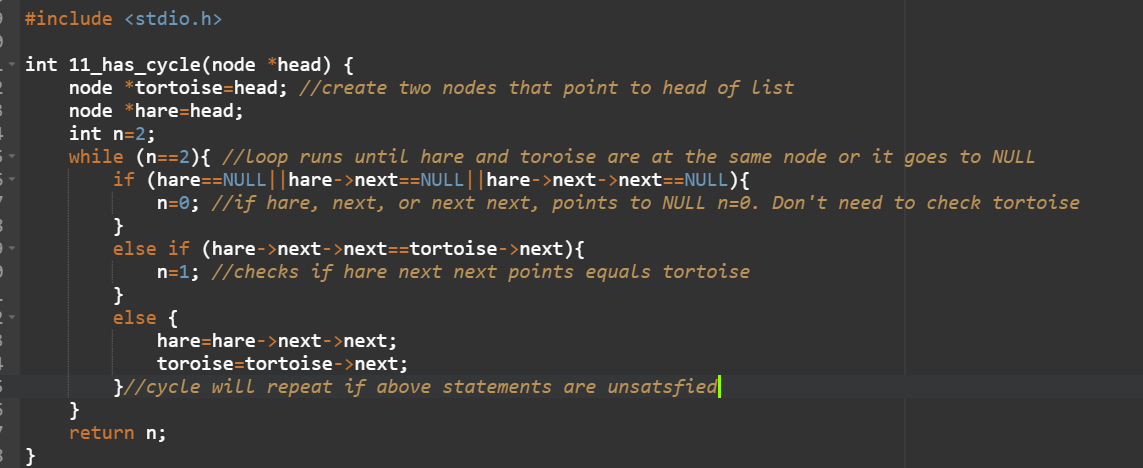
(a!=NULL) was changed to (a!=NULL && b!=NULL)

Fixed code:  
int ll\_equal(const node\* a, const node\* b) {  
 while (a!=NULL && b!=NULL) {  
 if (a->val != b->val)  
 return 0;  
 a = a->next;  
 b = b->next;  
 }  
 /\* lists are equal if a and b are both null \*/  
 return a == b;  
}

Exercise 4:

CGDB can run to completion by creating a separate .txt file, with the input “Andrew”. Next, we run the gdb on the interactive\_hello.c file and type “run < myname.txt” to run the program.

Exercise 5:



The code checks if a singly-linked list has a cycle. This code starts with two pointers at the head of the list called tortoise and hare. First, it moves the hare by two nodes. If it is not possible due to a NULL pointer the list ends and is acyclic. Next the tortoise is advance by one node. Then we check if the tortoise and hare are at the same node which means the list is cyclic.