

Module: R1: C Programming

Section: C Arrays, Pointers & Strings Task: 2.1

LAB 1: <https://github.com/ImNomanCR7/su21-lab-starter.git>

Exercise 1:

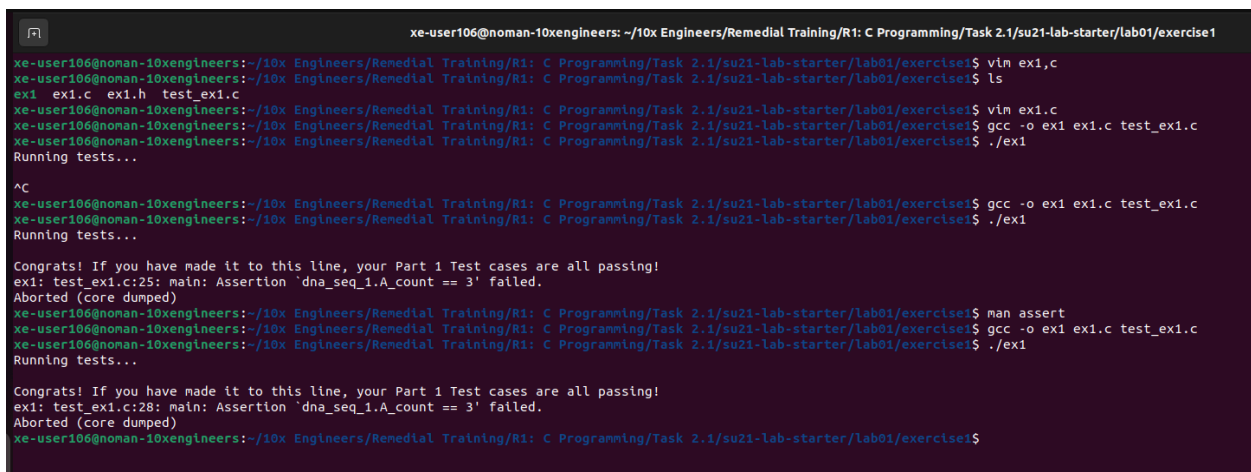
1. Number of Occurrences

Code Snippet:

```
int num_occurrences(char *str, char letter) {
    /* TODO: implement num_occurrences */
    int count = 0;
    while (*str != '\0'){
        if (letter == *str){
            Count++;
        }
        Str++;
    }
    return count;
}
```

Test Case:

```
//Noman's Test Case
int num_o = num_occurrences(str, 'o');
assert(num_o == 2);
```



```
xe-user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ vi ex1.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ ls
ex1  ex1.c  ex1.h  test_ex1.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ vi ex1.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ gcc -o ex1 ex1.c test_ex1.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ ./ex1
Running tests...

^C
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ gcc -o ex1 ex1.c test_ex1.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ ./ex1
Running tests...

Congrats! If you have made it to this line, your Part 1 Test cases are all passing!
ex1: test_ex1.c:25: main: Assertion 'dna_seq_1.A_count == 3' failed.
Aborted (core dumped)
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ man assert
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ gcc -o ex1 ex1.c test_ex1.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ ./ex1
Running tests...

Congrats! If you have made it to this line, your Part 1 Test cases are all passing!
ex1: test_ex1.c:28: main: Assertion 'dna_seq_1.A_count == 3' failed.
Aborted (core dumped)
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$
```

2. Nucleotide Occurrences

Code Snippet:

```
void compute_nucleotide_occurrences(DNA_sequence *dna_seq) {
    /* TODO: implement compute_nucleotide_occurrences */
    int count[4] = {0, 0, 0, 0};

    for (int i=0; i < strlen(dna_seq->sequence); i++){
        switch (dna_seq -> sequence[i]){
            case 'A':
                count[0]++;
                break;
            case 'C':
                count[1]++;
                break;
            case 'G':
                count[2]++;
                break;
            case 'T':
                count[3]++;
                break;
            default:
                break;
        }
    }

    dna_seq -> A_count = count[0];
    dna_seq -> C_count = count[1];
    dna_seq -> G_count = count[2];
    dna_seq -> T_count = count[3];
}
```

Test Case:

```
//Noman's Test case 2
DNA_sequence dna_seq_3;
strcpy(dna_seq_3.sequence, "AACGTACAGTTT");
```

```

compute_nucleotide_occurrences(&dna_seq_3);
assert(dna_seq_3.A_count == 4);
assert(dna_seq_3.C_count == 2);
assert(dna_seq_3.G_count == 2);
assert(dna_seq_3.T_count == 4);

```

```

xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ gcc -o ex1 ex1.c test_ex1.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ ./ex1
Running tests...

Congrats! If you have made it to this line, your Part 1 Test cases are all passing!
ex1: test_ex1.c:29: main: Assertion `dna_seq_1.A_count == 3' failed.
Aborted (core dumped)
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ gcc -o ex1 ex1.c test_ex1.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ ./ex1
Running tests...

Congrats! If you have made it to this line, your Part 1 Test cases are all passing!
Congrats! If you have made it to this line, your Part 2 Test cases are all passing!
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ gcc -o ex1 ex1.c test_ex1.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$ ./ex1
Running tests...

Congrats! If you have made it to this line, your Part 1 Test cases are all passing!
Congrats! If you have made it to this line, your Part 2 Test cases are all passing!
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise1$

```

Exercise 2:

1. Part 1 & Part 2

Removed compiler warnings at first and added `assert()` statements to debug which part of the code is causing problems, the updated code snippet after assertions is:

```

bool check_password(const char *first_name, const char *last_name, const
char
*password) {
bool length, upper, lower, number, name;
lower = check_lower(password);
assert(lower);
length = check_length(password);
assert(length);
name = check_name(first_name, last_name, password);
assert(name);
number = check_number(password);
assert(number);
upper = check_upper(password);
assert(upper);
return (lower && length && name && upper && number);
}

```

Output:

```

x@user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2
pwd_checker.c: In function 'check_name':
pwd_checker.c:75:21: warning: passing argument 1 of 'strstr' makes pointer from integer without a cast [-Wint-conversion]
75 |     const char *first = strstr(password, first_name);
    |                       ^~~~~~
    |                       |
    |                       char
In file included from pwd_checker.c:1:
/usr/include/string.h:350:34: note: expected 'const char *' but argument is of type 'char'
350 | extern char *strstr(const char *__haystack, const char *__needle)
    |                    ^~~~~~
x@user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ ls
pwd_checker  pwd_checker.c  pwd_checker.h  test_pwd_checker.c
x@user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc pwd_checker.c test_pwd_checker.c -o pwd_checker
x@user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ ./pwd_checker
Running tests...
pwd_checker: test_pwd_checker.c:12: main: Assertion 'test1' failed.
Aborted (core dumped)
x@user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc pwd_checker.c test_pwd_checker.c -o pwd_checker
Running tests...
pwd_checker: test_pwd_checker.c:12: main: Assertion 'test1' failed.
Aborted (core dumped)
x@user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc pwd_checker.c test_pwd_checker.c -o pwd_checker
pwd_checker.c: In function 'check_password':
pwd_checker.c:84:5: warning: implicit declaration of function 'assert' [-Wimplicit-function-declaration]
84 |     assert(lower);
    |     ^~~~~~
pwd_checker.c:3:11: note: 'assert' is defined in header '<assert.h>'; did you forget to '#include <assert.h>'?
2 | #include "pwd_checker.h"
+++ | #include <assert.h>
3 |
/usr/bin/ld: /tmp/ccz0l1n.o: in function 'check_password':
pwd_checker.c:(.text+0x202): undefined reference to 'assert'
/usr/bin/ld: pwd_checker.c:(.text+0x222): undefined reference to 'assert'
/usr/bin/ld: pwd_checker.c:(.text+0x24b): undefined reference to 'assert'
/usr/bin/ld: pwd_checker.c:(.text+0x26a): undefined reference to 'assert'
/usr/bin/ld: pwd_checker.c:(.text+0x289): undefined reference to 'assert'
collect2: error: ld returned 1 exit status
x@user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc pwd_checker.c test_pwd_checker.c -o pwd_checker
Running tests...
pwd_checker: pwd_checker.c:87: check_password: Assertion 'length' failed.
Aborted (core dumped)
x@user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc pwd_checker.c test_pwd_checker.c -o pwd_checker
Running tests...
pwd_checker: pwd_checker.c:88: check_password: Assertion 'length' failed.
Aborted (core dumped)

```

2. Part 3

After resolving the problem with assert length, another issue with assertion name comes up. I followed the same steps to identify and address the issue using cgdb.

After inspection, it appeared that the return value for the function **check_name** was not giving the required logic. Here's the corrected logic:

```

bool check_name(const char *first_name, const char *last_name, const
char *password) {
    /* Type "man strstr" in your terminal to learn what strstr does!
       To exit the man pages, press 'q' */
    /* Hint: a NULL pointer will evaluate to False in a logical
       statement while a non-NULL pointer
       will evaluate to True */
    const char *first = strstr(password, first_name);
    const char *last = strstr(password, last_name);
    return !(first || last);
}

```

```

xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc -g pwd_checker.c test_pwd_checker.c -o pwd_checker
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ ./pwd_checker
Running tests...

pwd_checker: pwd_checker.c:90: check_password: Assertion 'name' failed.
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc -g pwd_checker.c test_pwd_checker.c -o pwd_checker
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc -g pwd_checker.c test_pwd_checker.c -o pwd_checker
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ ./pwd_checker
Running tests...

pwd_checker: pwd_checker.c:92: check_password: Assertion 'number' failed.
Aborted (core dumped)
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$

```

3. Part 4

After removing the assert statements, I encountered an error with **check_number**. Upon inspection, it was revealed that we passed numbers 0-9 instead of the string '0' to '9'. Fixed that and recompiled the program.

```

bool check_number(const char *password) {
    while (*password != '\0') {
        if (check_range(*password, '0', '9')) {
            return true;
        }
        ++password;
    }
    return false;
}

```

Last bug which needs to be fixed is with **check_upper** function which is not behaving normally. Upon inspection, it was revealed that the bug originated from **check_range** function. Hence, it will take the following fix:

```

bool check_range(char letter, char lower, char upper) {
    bool is_in_range = (letter >= lower && letter <= upper);
    return is_in_range;
}

```

```

xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc -g pwd_checker
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ ls
pwd_checker  pwd_checker.c  pwd_checker.h  test_pwd_checker.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc -g pwd_checker
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc -g pwd_checker.c test_pwd_checker.c -o pwd_checker
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ ./pwd_checker
Running tests...

pwd_checker: pwd_checker.c:94: check_password: Assertion 'upper' failed.
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc -g pwd_checker
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc -g pwd_checker.c test_pwd_checker.c -o pwd_checker
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ ./pwd_checker
Running tests...

Congrats! The first test case is now passing. You should remove the assert statements that you added to pwd_checker.c because these correspond to the first test case and will not necessarily work for the
remaining test cases!

pwd_checker: pwd_checker.c:88: check_password: Assertion 'length' failed.
Aborted (core dumped)
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ vim pwd_checker.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ gcc -g pwd_checker.c test_pwd_checker.c -o pwd_checker
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$ ./pwd_checker
Running tests...

Congrats! The first test case is now passing. You should remove the assert statements that you added to pwd_checker.c because these correspond to the first test case and will not necessarily work for the
remaining test cases!

Congrats! You have passed all of the test cases!
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise2$

```

Exercise 3:**1. reverse_list SIGSEGV:****Code Snippet:**

```
void reverse_list(struct Node **head) {
    if (head == NULL || *head == NULL) {
        return;
    }
    struct Node *curr = *head;
    struct Node *next = (*head)->next;
    curr->next = NULL;
    while (next != NULL) {
        struct Node *temp = next->next;
        next->next = curr;
        curr = next;
        next = temp;
    }
    *head = curr;
}
```

```
xe-user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programmi...
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-s
tarter/lab01/exercise3$ ls
linked_list.c linked_list.h test_linked_list.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-s
tarter/lab01/exercise3$ gcc -g linked_list.c test_linked_list.c -o linked_list
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-s
tarter/lab01/exercise3$ ls
linked_list linked_list.c linked_list.h test_linked_list.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-s
tarter/lab01/exercise3$ ./linked_list
Running tests...

Segmentation fault (core dumped)
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-s
g/Task 2.1/su21-lab-starter/lab01/exercise3$ cgdb linked_list
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-s
tarter/lab01/exercise3$ gcc -g linked_list.c test_linked_list.c -o linked_list
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-s
tarter/lab01/exercise3$ ./linked_list
Running tests...

Congrats! You have passed the reverse_list test!

Segmentation fault (core dumped)
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-s
tarter/lab01/exercise3$
```

2. add_to_back SIGSEGV:**Code Snippet:**

```
void add_to_back(Node **head, int data) {
    if (head == NULL) {
        return;
    }
```

```

    }
    Node *new_node = create_node(data);
    if (*head == NULL){
        *head=new_node;
        return;
    }

    Node *prev;
    for (Node *curr = *head; curr != NULL; curr = curr->next) {
        prev = curr;
    }
    prev->next = new_node;
}

```

```

se-user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise3$ gcc -g linked_list.c test_linked_list.c -o linked_list
se-user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise3$ ./linked_list
Running tests...
Congrats! You have passed the reverse_list test!
Congrats! All of the test cases passed!
se-user106@noman-10xengineers: ~/10x Engineers/Remedial Training/R1: C Programming/Task 2.1/su21-lab-starter/lab01/exercise3$

```

Exercise 4:

1. Cyclic Linked-Lists:

Code Snippet:

```

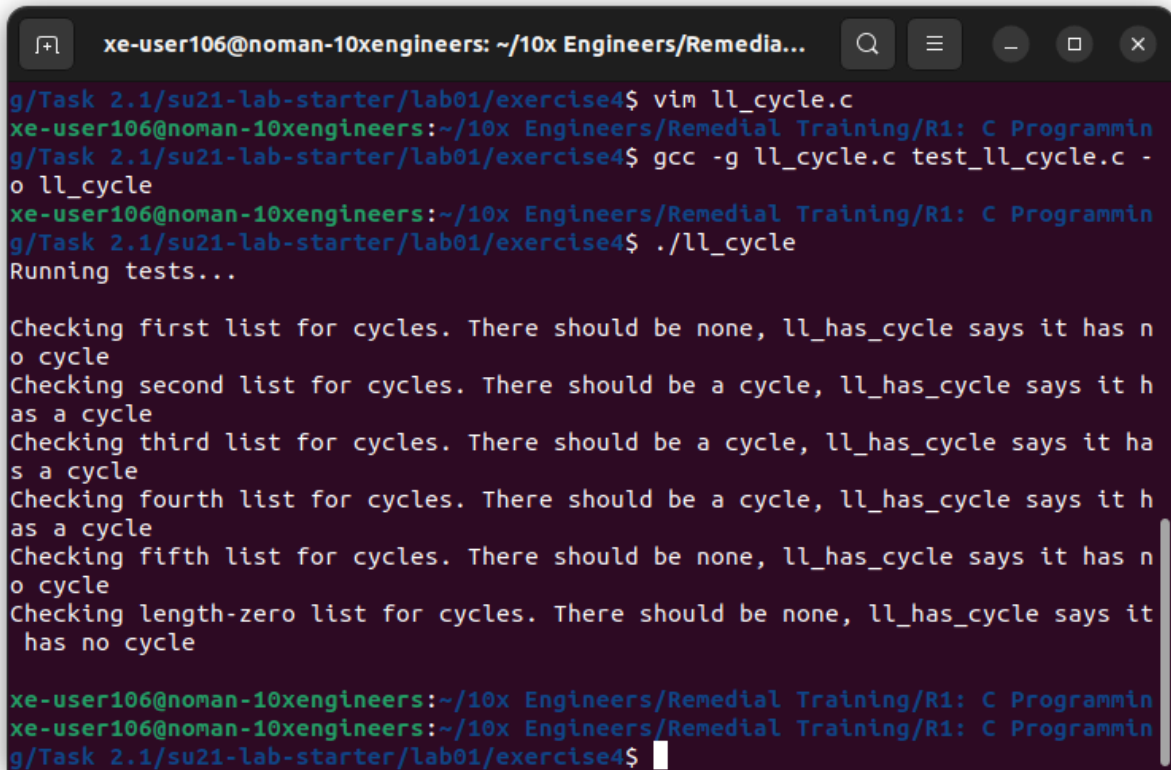
#include <stddef.h>
#include "ll_cycle.h"

int ll_has_cycle(node *head) {
    /* TODO: Implement ll_has_cycle */
    node *fast_ptr = head;
    node *slow_ptr = head;

    while (fast_ptr != NULL && fast_ptr->next != NULL){
        fast_ptr = fast_ptr->next->next;
        slow_ptr = slow_ptr->next;

        if (fast_ptr == slow_ptr){
            //Has a Cycle
            return 1;
        }
    }
    //Acyclic
    return 0;
}

```



```
xe-user106@noman-10xengineers: ~/10x Engineers/Remedia...
g/Task 2.1/su21-lab-starter/lab01/exercise4$ vim ll_cycle.c
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programmin
g/Task 2.1/su21-lab-starter/lab01/exercise4$ gcc -g ll_cycle.c test_ll_cycle.c -
o ll_cycle
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programmin
g/Task 2.1/su21-lab-starter/lab01/exercise4$ ./ll_cycle
Running tests...

Checking first list for cycles. There should be none, ll_has_cycle says it has n
o cycle
Checking second list for cycles. There should be a cycle, ll_has_cycle says it h
as a cycle
Checking third list for cycles. There should be a cycle, ll_has_cycle says it ha
s a cycle
Checking fourth list for cycles. There should be a cycle, ll_has_cycle says it h
as a cycle
Checking fifth list for cycles. There should be none, ll_has_cycle says it has n
o cycle
Checking length-zero list for cycles. There should be none, ll_has_cycle says it
has no cycle

xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programmin
xe-user106@noman-10xengineers:~/10x Engineers/Remedial Training/R1: C Programmin
g/Task 2.1/su21-lab-starter/lab01/exercise4$
```

Repo URL: <https://github.com/ImNomanCR7/su21-lab-starter.git>