Linked List and Array List

To testers:

Please first make sure the student's "main_ll.c" and "main_al.c" is the same as the two provided on page 4 and 5. Then run the following four commands (shown in the script) "make clean", "make runboth", "valgrind ./main_al", "valgrind ./main_ll". (My shell prompt starts with an arrow.)

```
➜ make clean
➜ make runboth
gcc -c main_ll.c
gcc -c ll.c
gcc -o main_ll main_ll.o ll.o
gcc -c main_al.c
gcc -c al.c
gcc -o main_al main_al.o al.o
./main_ll
H -> E -> L -> L -> O -> NULL
H -> A -> E -> L -> L -> O -> NULL
K -> H -> A -> E -> L -> L -> O -> NULL
K -> H -> A -> E -> L -> L -> O -> O -> NULL
X -> K -> H -> A -> E -> L -> L -> O -> O -> NULL
===removing===
K -> H -> A -> E -> L -> L -> O -> O -> NULL
H -> A -> E -> L -> L -> O -> O -> NULL
H -> A -> L -> L -> O -> O -> NULL
H -> A -> L -> L -> O -> NULL
./main_al
[ H | E | L | L | O ]
[ H | A | E | L | L | O ]
[ K | H | A | E | L | L | O ]
[ K | H | A | E | L | L | O | O ]
[ X | K | H | A | E | L | L | O | O ]
===removing===
[ K | H | A | E | L | L | O | O ]
[ H | A | E | L | L | O | O ]
```

```
[ H | A | L | L | O | O ]
[ H | A | L | L | O ]
➜  valgrind ./main_al
==43991== Memcheck, a memory error detector
==43991== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==43991== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==43991== Command: ./main_al
==43991==
[ H | E | L | L | O ]
[ H | A | E | L | L | O ]
[ K | H | A | E | L | L | O ]
[ K | H | A | E | L | L | O | O ]
[ X | K | H | A | E | L | L | O | O ]
===removing===
[ K | H | A | E | L | L | O | O ]
[ H | A | E | L | L | O | O ]
[ H | A | L | L | O | O ]
[ H | A | L | L | O ]
==43991==
==43991== HEAP SUMMARY:
==43991==     in use at exit: 0 bytes in 0 blocks
==43991==   total heap usage: 2 allocs, 2 frees, 1,128 bytes allocated
==43991==
==43991== All heap blocks were freed -- no leaks are possible
==43991==
==43991== For counts of detected and suppressed errors, rerun with: -v
==43991== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
➜  valgrind ./main_ll
==43996== Memcheck, a memory error detector
==43996== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==43996== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==43996== Command: ./main_ll
==43996==
H -> E -> L -> L -> O -> NULL
H -> A -> E -> L -> L -> O -> NULL
K -> H -> A -> E -> L -> L -> O -> NULL
K -> H -> A -> E -> L -> L -> O -> O -> NULL
X -> K -> H -> A -> E -> L -> L -> O -> O -> NULL
===removing===
K -> H -> A -> E -> L -> L -> O -> O -> NULL
```

H -> A -> E -> L -> L -> O -> O -> NULL
H -> A -> L -> L -> O -> O -> NULL
H -> A -> L -> L -> O -> NULL
==43996==
==43996== HEAP SUMMARY:
==43996== in use at exit: 0 bytes in 0 blocks
==43996== total heap usage: 10 allocs, 10 frees, 1,168 bytes allocated
==43996==
==43996== All heap blocks were freed -- no leaks are possible
==43996==
==43996== For counts of detected and suppressed errors, rerun with: -v
==43996== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

main_ll.c

```
#include <stdlib.h>
#include <stdio.h>
#include "linked_list.h"

int main(void) {
  char ls[] = "HELLO";
  node_t *ll2 = make_list(ls, 5);
  print_list(ll2); // HELLO

  ll2 = insert_at(ll2, 'A', 1); // HAELLO
  print_list(ll2);
  ll2 = insert_at(ll2, 'K', 0); // KHAELLO
  print_list(ll2);
  ll2 = append(ll2, 'O'); // KHAELLOO
  print_list(ll2);
  ll2 = prepend(ll2, 'X'); // XKHAELLOO
  print_list(ll2);

  printf("===removing===\n");

  ll2 = remove_ith(ll2, 0); //KHAELLOO
  print_list(ll2);
  ll2 = remove_ith(ll2, 0); //HAELLOO
  print_list(ll2);
  ll2 = remove_ith(ll2, 2); //HALLOO
  print_list(ll2);
  ll2 = remove_ith(ll2, 4); //HALLO
  print_list(ll2);

  return (0);
}
```

main_al.c

```
#include <stdlib.h>
#include <stdio.h>
#include "array_list.h"

int main(void) {
  char ls[] = "HELLO";
  array_list_t *ll2 = make_list(ls, 5);
  print_list(ll2); // HELLO

  insert_at(ll2, 'A', 1); // HAELLO
  print_list(ll2);
  insert_at(ll2, 'K', 0); // KHAELLO
  print_list(ll2);
  append(ll2, 'O'); // KHAELLOO
  print_list(ll2);
  prepend(ll2, 'X'); // XKHAELLOO
  print_list(ll2);

  printf("===removing===\n");

  remove_ith(ll2, 0); //KHAELLOO
  print_list(ll2);
  remove_ith(ll2, 0); //HAELLOO
  print_list(ll2);
  remove_ith(ll2, 2); //HALLOO
  print_list(ll2);
  remove_ith(ll2, 4); //HALLO
  print_list(ll2);

  return (0);
}
```

ENGR120 Chapter 13 Test Results

Student Name: _____

Date: _____ Time: _____

Tester (Do not sign if the Student Name is blank):_____

Linked List and Array List

Code compiles:☐Y ☐N # of warnings: _____
Does not compile only because of Makefile: ☐Y ☐N
==========
Code ran: ☐Y ☐N
Terminated OK: ☐Y ☐N
Output Correctly (Linked List Implementation): ☐Y ☐N
Comment:
Output Correctly (Array List Implementation): ☐Y ☐N
Comment:
Output was free from extraneous output: ☐Y ☐N
==========

valgrind check – no memory leak: ☐Y ☐N
If has leak, how much:

valgrind check – no read/write violation: ☐Y ☐N
If has leak, how much:

==========
Comments:

Grading Rubric

		% usually deducted
A.	Defects in source code not revealed by tests	10-30
B.	Handed in extra source files	10
C.	Names are not meaningful	10
D.	Indentation does not indicate program structure	10
E.	Program will not compile	50-100
F.	Program produces incorrect results	20–100
G.	Incorrect implementation	20-100
H.	Does not compile only because of Makefile	10
I.	Program solves wrong problem	10–100
J.	Use of unnamed constant	10

Assigned : Linked list and array list implementations. Hand in “ll.c”, “al.c” only in hard-copy.

Name: _____ Points: _____ / 50 total

Item	Lists		
Tutor Report:	□ Tests OK (E) (F)	Points:	
Tutor Comment:			
□ Includes purpose comment (A)	□ Adequate commenting (B)	□ Meaningful names (C)	□ Indentation (D)
□ Use of #defines / constants (K)	□ Clean output (H)	□ Uses functions (L)	
□ Evidence of test cases (G)	□ Algorithm design (J)	□ Shows digits	□ Uses integers
Comments:			