TESTING OF ASSIGNMENT 4

1 Overview

The assignment asked you to code several functions that manipulate singly linked lists. The linked list in this case is of particular type set of data fields (first name, family name and id). The testing of the functions is carried out one function at a time.

2 Automated testing

The automated testing will be carried using two test programs. One program is testing the basic functionality of the different functions, excluding the deleteList() and the printList() functions. The second program is testing the latter two functions and memory leaks that may occur during the delete step. Here we also test the insertion by allowing the program to create a controlled memory leak of nodes. Namely, the program will insert nodes but not delete them.

The automation is carried out by compiling the your code and linking it with the test programs.

2.1 Test 1

The first test suite tests each function (in most cases) on: a. an empty list; b. on a list that has a single node; and, c. a list that has multiple nodes.

The set of tests is designed in such a way that if the function causes a segmentation violation (program crashes) then the problem would not affect the overall test program. This is achieved by cloning the test program (using the fork() function) and then invoking the test function.

This set of test examines:

a. Insertion – here the program tests three insertion functions: insertToList(), insertAfter() and insertLast()
b. Search – here the program tests two search functions: serachById() and searchByName()
c. Deletion – here the program tests four delete functions: deleteFromList(), deleteNodeByName(), deleteLast() and deleteAfter()
d. Miscellaneous functions – here the test program tests three functions: countRecords(), listSize() and removeDuplicates().

2.1.1 Generating the test program

The test program is created by linking the student's files and the test program.
The required files are:

1. Files linked_list.h and linked_list.c – these are the files that you coded as part of the assignment.
2. File main_test.o – this is an object file that contains the test program.
3. Makefile – this is the make file that would generate the program.

Program generation steps:

a. Download the your progam files
b. Ensure that the files main_test.o and Makefile are in the same directory as your files
c. Create the test program by calling make (note that here we use Makefile which is the default file)
d. The output should be an executable program main_test

Note, that there can be some issues:

a. Unresolved externals – it may be that you did not implement some of the functions or changed the names. If you changed the function names then the program will not link and unresolved eternals error will be displayed. If you did not implement some of the functions then the program should still link because the assignment file contains stabs for each function.
   Solution: add the missing functions.

2.1.2 Program testing and grading

Invoke the program main_test and test the code. Note that after each step the test program determines that maximum points that the student code earned to a maximum total of 79.

2.2 Test 2

The second test suite tests memory leaks, the print list function and the delete list function. It will check the insert functions, the printList() and the deleteList() functions on: a. an empty list; b. a list that has multiple nodes.

The set of tests is designed in such a way that if the function causes a segmentation violation (program crashes) then the problem would not affect the overall test program. This is achieved by cloning the test program (using the fork() function) and then invoking the test function. Since Test 2 is testing memory leaks the program morphs itself to a valgrind program.

This set of test examines::

a. Printing – here the program tests one function: printList().
b. Deletion – here the program tests five delete functions: deleteFromList(), deleteNodeByName(), deleteLast(), deleteAfter()and deleteList()
c. Insertion – here the program insert records to the list but does not delete the list. As a result there is a controlled memory leak. The program should display the amount of expected memory leak.

2.2.1 Generating the test program

The test program is created by linking the student's files and the test program.
The required files are:

1. Files linked_list.h and linked_list.c – these are the files that you coded as part of the assignment.
2. File main_test_delete_leak.o – this is an object file that contains the test program that tests the leaks. This program is invoked by valgrind.
3. File main_test_delete – this is the executable program of this suits. You do not need to create it. It is already available
4. Makefile_leak – this is the make file that would generate the program main_test_delete_leak.

Program generation steps:

a. Download the files
b. Ensure that the files main_test_delete_leak.o or main_test_delete_leak.c and Makefile_leak are in the same directory as our files
c. Create the test program by calling make –f Makefile_leak
d. The output should be the executable program main_test_delete_leak

2.2.2 Program testing

Invoke the program main_test_delete and test the your code. Note that the program will clone itself and morph into a valgrind program. So test may take a few more seconds. The purpose of testing is to ensure that no memory leaks occur.

The test program is invoking valgrind. Therefore, you will need to check at the end of each test whether there was a memory leak. Note that in case of the insertion testing there is a controlled memory leak. After each insertion test the test program should inform you what is the expected memory leak.