CS 451
Programming Assignment #1,2
Imperative programming: C
Abstract Data Types
Due 9/16, 9/21/16, by 6:00am sharp!

Introduction

The goal of this assignment is to get further acquainted with the C programming language. There are two parts to this assignment: the writing of a linked list abstract data type (ADT) and the writing a hash table ADT that delegates much of its functionality to the linked list.

Part 1:

Pervasive idea: the list should work for any element data type

Write a linked list ADT that supports the following operations: list_find, list_foreach, list_foreach1, list_foreach2, list_free, list_initialize, list_insert_beginning, list_insert_end, list_isempty, list_member, and list_remove. If you have any questions on what a particular operation does ASK!

Your implementation must use and not modify the main program and header files found on the course web page (i.e., main.c, and list.h). You must use my main program because I will use its interface to grade your program. Finally, you can modify and must use the makefile from the course web page.

What to hand in

(1) Email me the source as a single attachment named <your name>-a1.c by 6:00am on the due date.
   Your program’s first line must be “//This is my code.”
(2) A copy of the executable — named list — to your home directory. Runnable by me!

Notes

Consider first writing a list of ints and then a list of students. Then consider where they are identical and how you might abstract out the remaining functionality.

Part 2:

Pervasive idea: the hash table should work for any element data type

If you implemented Part 1 correctly, this part should be trivial :) Our hash table will have seven parts: an array of “buckets”, a constructor, a destructor, three access functions, and a collection of iterators. A bucket is simply a pointer to a linked list. All values having the same “hash value” are stored in the same linked list. Hash values are computed from the data records to be stored in the hash table by a hash function.

The constructor function ht_initialize allocates space for an empty hash table, initializes its size, hash function, and comparison function fields, and initializes the each bucket (list). The destructor function ht_free frees all memory currently allocated to the hash table. The three access functions ht_delete, ht_insert, and ht_lookup perform the obvious operations. Each iterator, ht_foreach?, (the “?” indicates the number of additional parameter the function takes) takes a
hashtable, a function, and zero to two additional arguments; it then calls the function with each element in the hashtable and the additional arguments.

Your implementation must use and not modify the main program and header files found on the course web page (i.e., main.c, and ht.h). You must use my main program because I will use its interface to grade your program. Finally, you can modify and must use the makefile from the course web page.

What to hand in

(1) Email me the source as a single attachment named `<your name>-a2.c` by 6:00am on the due date.
   Your program’s first line must be “//This is my code.”
(2) A copy of the executable — named `hash` — to your home directory until the project is graded.

Notes

(1) Expect to do more thinking than coding. My solution is only 50 executable lines of code.

Security

You should secure your source code. I suggest the following:
After logging issue the following commands from your home directory
   `mkdir cs451`
   `mkdir cs451/assignment.1`
   `mkdir cs451/assignment.1/part1`
   `mkdir cs451/assignment.1/part2`
   `chmod 700 cs451 cs451/assignment.1 cs451/assignment.1/part1 cs451/assignment.1/part2`
   `<download the necessary files form the course web page>`

To work on your program issue the following commands
   `cd cs451/assignment.1`
   `<develop your program>`

If this is unclear please read the man pages for chmod and mkdir or come to talk to me.