

---

**CS 451**  
**Programming Assignment #1,2**  
**Imperative programming: C**  
**Abstract Data Types**

Due 9/11,16/08, by 6:00am sharp!

**Introduction**

The goal of Assignments 1 and 2 is to get further acquainted with the C programming language. Assignment 1 is to write list abstract data type (ADT); Assignment 2 is to write vector and queue ADTs that delegate much of their work to the list.

**Pervasive idea:**  
**an ADT should work for *any* data type**

**Assignment 1:**

Write a (linked) list ADT . A list is an ordered container that supports the following functionality: a constructor, a destructor, and eight access functions. The constructor function `list_initialize` allocates an empty queue and initializes its internal state. The destructor function `list_free` frees all memory currently allocated by list ADT code. Each of the eight access functions `list_find`, `list_foreach`, `list_foreach1`, `list_insert_beginning`, `list_insert_end`, `list_isempty`, `list_member`, and `list_remove` perform the obvious operation. 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 by 9/11/08**

- (1) Email the source for `list.c` (only!) by 6:00am on the due date.  
(try something like `mail -s 451asn1 binkley < list.c`.)  
Your program's first line **must** be `///This is my code.`
- (2) A copy of the executable—named `list`—it left in your home directory until the project is graded.

**Notes**

- (1) Before you think about modifying files other than `list.c`, you must first get my permission!
- (2) Use loooong names (at least ten characters) [-1 point per missing letter :)]
- (3) Do — have function headers, use `free`, check for error returns, free only what you allocate.
- (4) Don't — generate output from within an ADT.
- (5) The -1 point per missing letter is an idle threat.
- (6) Proper language style - by convention, C variables are written with underscores not using camel case. Just do it!

continued over leaf

## Assignment 2:

### Pervasive idea: the queue and vector should work for *any* element data type

If you implemented Assignment 1 correctly, implementing the two ADTs of this assignment should be trivial :) First, a queue is a first-in, first-out container (I'll let you choose the representation) that supports the following functionality: a constructor, a destructor, three access functions, and for testing, a print function. The constructor function `queue_initialize` allocates an empty queue and initializes its internal state. The destructor function `queue_free` frees all memory currently allocated by queue ADT code. The three access functions `queue_enqueue`, `queue_dequeue`, and `queue_front` perform the obvious operations. Finally, `queue_print` prints the contents of the queue from front to back.

Second, a vector is an  $n$ -tuple that supports the following functionality: a constructor, a destructor, three operations, and for testing, a print function. The constructor function `vector_initialize` allocates an empty vector and initializes its internal state. The destructor function `vector_free` frees all memory currently allocated by vector ADT code. The three operations `vector_add`, `vector_scale`, and `vector_dot_product` perform the obvious operations. To implement `vector_dot_product` requires adding a new functionality to your list. This one change to the list is permitted without asking. Finally, `vector_print` prints the contents of the vector.

Your implementations must use the main programs and header files found on the course web page. You must use my main programs because I will use their interfaces to grade your programs. Finally, you can modify and must use the `Makefiles` on the course web page.

### What to hand in by 9/16/08

- (1) Email the source for `queue.c` and `vector.c` by 6:00am on the due date. Your program's first line **must** be `//This is my code.`
- (2) A copy of the executables—named `vector` and `queue`—it left in your home directory until the project is graded.

### Notes

- (1) Read the notes for Assignment 1.
- (2) Expect to do more thinking than coding. My solutions are only 37 and 45 executable lines of code, respectively.

---

## Security

You should secure your source code. I suggest the issuing the following commands after logging:

```
cd ~
mkdir cs451
mkdir cs451/assignment.1
mkdir cs451/assignment.2
chmod 700 cs451 cs451/assignment.1 cs451/assignment.2
cd cs451/assignment.1
< download the necessary files for assignment 1 >
```

To work on Assignment 1, issue the following commands

```
cd ~/cs451/assignment.1
< develop your program >
```

To place the final copy of your executable hash table in your home directory issue the following commands

```
cp <executable program> ~/list
chmod 711 ~/list
```

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

## Learn from your predecessors mistakes!

- remove tabs before handing in code. In vi use “:1, \$s/^V^I/ /g” assuming you have 4 space tab stops.
- use “^L” to insert page breaks.
- put functions in alphabetic order
- don’t comment the obvious
- don’t forget function header comments
- aim for declarative comments (not operational) (e.g., "delete X" rather than "deletes X")
- omit “a pointer to” in your descriptions.
- watch for non-void functions that fail to return a value
- don’t have #defined functions without parameters
- no printing from inside ADT
- don’t use globals unless you know what you are doing (you don’t :)).
- use for loops.
- ack, awk, naming
- don’t assuming true == 1
- **testing** - past programs broke on simple inputs such as the empty list.
- do your memory allocation (malloc, free) homework.  
(e.g., malloc NULL returns checks; free what you allocate and ONLY what you allocate)
- Think about booleans as values:  
“if X then return TRUE else return FALSE” is simply “return X”
- protection:  
I must be able to run the executable, I must NOT be able to access your code!