General Rules

- Due date is October 16, 2006, Monday and not subject to postpone.
- Late submission is allowed with a 10% deduction penalty per day.
- There is no teaming up. The homework has to be done/turned in individually. In case of cheating, all parts involved (source(s) and receiver(s)) get zero.
- Please follow the webpage regularly because there may be additional explanations for the homework.
- You will submit your homework electronically. The details about the submission process will be announced in the course web page.

Homework

The main purpose of this homework is to review your C knowledge and to learn about pointers, structures and structure arrays. In this assignment you are required to implement a CD stock for a store. The store wants to keep for each CD, its number (int), title (char [20]), composer (char[15]) and a count (int) of that CD in stock. Each CD should have a unique number.

Write a program that implements this stock information. The stock information is actually going to be stored in a structure array.

You should present the user a menu that allows him/her to do the following operations:

1) Stock a new CD
2) Sell a CD
3) Print the details of a CD
4) Quit

When the user selects the first menu option, your program must ask the following information of the new CD:
- CD number
- CD title
- Composer name
- The amount to be stocked.

If the CD already exists in the stock your program should not insert it again. You should check the uniqueness of the CD number.

When the user selects the second menu option, your program should ask the CD number, search the list and if it is found, it should decrease the amount in stock. If the amount is zero, then an appropriate message must be printed. If the CD is not in stock, then print an appropriate message.

When the user selects the third menu option your program should ask the number of the CD and print the details of it (i.e. the number, title, composer, count) if it exists in stock.
When the user selects the fourth menu option, you quit from the program.

After executing a menu choice, the menu should keep on popping up again until the user chooses to quit.

**Implementation Requirements**

This program is somewhat open ended so that you can interpret some things for yourself and decide how to implement them. But there are some restrictions on your design:

1) Create a CD_type structure having the following fields:
   - CD number
   - CD title
   - Composer
   - CD count.

2) Use an array of CD_type structure.

3) Write a function to read the data of a CD from the keyboard. Use the following function prototype:

   ```c
   void GetData(CD_type * cd);
   ```

4) Write a function to insert a CD into the CD array. Use the following function prototype where cd is a pointer to a CD structure (to implement call-by-reference), List is the CD array and n is the number of CDs in the array passed as call-by-reference again. After insertion the number of elements must be increased by one.

   ```c
   void insertCD(CD_type * cd, CD_array List[], int * n);
   ```

5) Write a function to search a CD in the CD Array. The function takes the CD number, the CD array and the effective size of the array as its parameters and returns the subscript of the location where the CD is found. If the CD is not in the array the function returns –1. Use the following function prototype:

   ```c
   int searchCD(int CD_number, CD_array List[], int n);
   ```

6) Write a function to print the details of a CD. The function takes the subscript of the element in the CD array and prints the contents of that element. Use the following function prototype:

   ```c
   void printDetails(int index, CD_array List[]);
   ```

7) Write a function to decrease the amount in stock when a CD is sold. Note that if the amount is already zero, the function does not change anything, but prints an appropriate message. The function takes the array subscript of the location of the CD and the CD array as its parameters. Use the following function prototype:

   ```c
   void decreaseStock(int index, CD_array List[]);
   ```

When writing your program use the topdown modular design approach. Some modules may be used more than once. For example, the function searchCD may be used in several places: before a call to printDetails, or before a call to decreaseStock etc. Think about the modular design before writing the code.
Furthermore a correct program should not crash if it does not get the exact input it expects. Thus it is left to you to figure out what kind of error checking is necessary so that the program does not crash.