CSc 3380 Assignment1 (**15 pts)** Spring 2016

Due:  **February 2, Tuesday 8:00 AM**

**Student Performance Objectives:**

* To review structured programming (Sequence, Selection, Repetition- NO BREAK, GOTO, CONTINUE)
* To review modular design (methods [functions] that call methods [functions] help solve the problem of the calling method [function])
* To review the array-based implementation of the list

**Problem Scenario:**

Groovy Music , Inc. has requested a program to help maintain the inventory of music DVDs. Groovy Music has limited shelf space in the store and therefore the manager must remove all obsolete DVDs from the display shelves on a yearly basis. The program should request the cut-off year from the program user. All DVDs with a release date *below* and *equal* to the cut-off date will be pulled from the shelves.

The program should print a report with three sections:

#1. Print the inventory in original order by the DVD identification number

#2. Print detailed information of the DVDs to be removed from the shelves

#3. Print the final updated inventory report (omitting the DVDs to be removed)

For each DVD, show the inventory value calculated as the inventory quantity times the current selling price. Also the overall value of the total inventory should be calculated and shown as summary information for all three report sections.

**Assignment Requirements:**

1. The program must include the **required documentation** and follow programming documentation guidelines.
   1. Include **Identification Header, Problem Specification**, **Problem Analysis, Implementation Directions:**  as comments at the beginning of your program.
      1. **IDENTIFICATION HEADER:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Groovy Music End-of-Year Inventory \*/

/\* FirstName LastName \*/

/\* CSC 3380\_XX Section 1 \*/

/\* Assignment 1 \*/

/\* Instructor: Douglas \*/

/\* Program due date: February 2, 2016 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

* + 1. Describe the **PROBLEM SPECIFICATION** as a one line infinitive statement.
    2. Provide the **PROBLEM ANALYSIS**. In a few brief sentences describe the inputs, the outputs and any formulas that are needed to solve the problem.
    3. **IMPLEMENTATION DIRECTIONS** must include specific instructions on how to compile the program and how to execute the program.
  1. Use good documentation:

Use self-documenting variable names.

Comment all significant blocks (i.e. cohesive related statements).

Comment instructions (action statements) *before* the statement(s) described – *not after*

Comment instructions on a line by itself, *never* on the same line as the code statement.

Use the same indentation for comments as for the code.

Use a consistent number of spaces in which to indent the program.

Indent the entire program by 3 spaces to avoid printing in the gutter

and so that debug statements are easily recognized for removing before submission.

Use blank lines to set-off significant code blocks. (**Vertical Spacing).**

1. The program must request the cut-off year from the user. Verify that the year is between 2006 and 2016 inclusive. User should be prompted to repeatedly supply the input data until the cut-off year is with the range.
2. **INPUT FILE:**  ASCII text file, **Music2015.data**  
   All data fields are separated by spaces. The file may be in random order.  
   *For each CD record there are the following data values separated by spaces:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Line 1* | CD Identification  (integer) | Music Category Code  (1,2,3, or 4) | Release Year (integer) | ‘\n’ |

|  |  |  |  |
| --- | --- | --- | --- |
| *Line 2* | Selling Price  (real) | Inventory Count (Balance on Hand)  (integer) | ‘\n’ |
| *Line 3* | Artist Name  (string – **spaces included**) | | ‘\n’ |
| *Line 4* | CD Title  (string – **spaces included**) | | ‘\n’ |

1. Data should be described using a CD\_dataclass(structure) containing the ID, Category, Release Year, …, Artist Name, Title.
2. Music categories are:

Music Categories:

1. “”Blues”
2. “Classical”
3. “Country”
4. “Folk”
5. **Array-based** implementation of the linear list ADT (Abstract Data Type):
   * NO NOT USE library code for the ADT!
   * A ***class*** must be used to maintain the list ADT. The music data is stored as a private   
     array (consecutive memory locations).
   * Input data should be read and stored into the list, until end of file is detected. Use the ADT **insert** operation to fill the list from the input file. There is a maximum of 20 DVDs in the inventory. [Note that a special input system call may be used to read the entire line for strings including spaces. Investigate alternatives.]
   * Create a special **linear-ordered traversal** operation to walk through the list and to determine if the DVD is to be dropped from the inventory. If a DVD is to be dropped from the inventory, utilize the ADT **delete** operation.
6. Do not create a new output file from the updated list. See note below.
7. **OUTPUT FILE**: ASCII text file **Report.out**Use appropriate headings and good columnar layout for the report sections. *Don’t forget the requested summary information.*
   * (#1) List of the inventory contents before deletions
   * (#2) List of the inventory items to be deleted.
   * (#3) List of the inventory contents after deletions

*NOTE: This program presents a convoluted way of solving the problem. In the real-world, an update program would simply read the input file (Music2015.data) and for each record test against the cut-off date. A new files would be generated: Music2016\_Drop.data and Music2016.data. Then a report program would be run to print each file, Music2015.data (before), Music2016\_Drop.data (removed from the shelves), Music2016.data (new inventory data file). Creating the new file versions without altering the old ones creates a good audit trail. However, this solution demonstrates a simple illustration of the ADT.*

### Using UNIX

**Part 1**. Logon to UNIX. From your home directory, create the prog1 subdirectory and change to the prog1 subdirectory.

**>mkdir prog1  
 >cd prog1**

**>pwd**

### Part 2. Implementation

### Create your files, compile, execute, test.

1. To end this session: **>logout**

**Part 3. Electronic Submission. *DO NOT resubmit*** after the deadline!!!   
1. Go to your home directory(verify with >pwd).

Copy your prog1 subdirectory to the grader's directory:

>**~cs3380\_dou/bin/p\_copy 1**

# 2. Make sure the copy was accomplished by using the verify command to list the number of files copied: >~cs3380\_dou/bin/verify 1

# 