**Album Management System**

You are requested to develop **Album** **Management System**. The System keeps information about an artist and his/her albums.

**Information About Artist:**

* artistId: unique id of the artist(auto generated with prefix art-)
* name: Full name of artist
* gender: Gender of artist
* phone: Phone of Artist
* email: Email of Artist

**Information About Album:**

* albumId: unique Id of album(auto generated integer with prefix alb-)
* artistId: Id of artist(references id in artist information)
* title: Title of publication or album name
* format: Format of recording cd, ..,
* date: date published
* Path: The path where the album is located

**Requirement**

* The system must allow the user to add artist
* The system should allow the user to edit artist
* The system should allow the user to view artist info
  + View All
  + View by search(Search is prefix search)
* The system should allow the user to add album for an artist
* The system should allow the user to edit album information
* The system should allow the user to remove albums of artist

The break down and interface definition is already done for you (look below)

**Important**

* The assignment is to be done with a group of three students
* Submission will be done in three versions
  + Version 1(with no dynamic memory and file)

Submission Date: After two week

* + Version 2(with dynamic memory )

Submission Date: To be announced

* + Version 3(with file )

Submission Date: To be announced

* Defense dates: To be announced
* Deadline must be respected
* Project must be done according to the specification
* Each group should submit its own work. Duplicate work receive no credit
* Each member must participate actively

**Structure Chart**

Note needed for version 1 and 2

main

loadding

mainManager

farewell

loadAlbum

loadArtist

welcome

openFile

sortAlbum

openFile

sortArtist

mainMenu

albumManager

artistManager

displayOneArtist

editArtistInfo

selectArtist

addArtist

deleteArtist

editArtist

artistEditor

artistMenu

geArtistInfo

displaySearchResult

artistViewer

editArtistMenu

viewArtistBySearch

displayAllArtist

viewArtistMenu

displaSearchArtistResult

searchArtist

sortArtist

SortArtist

searchArtist

SearchArtistByName

searchArtistById

getArtistName

getArtistGender

getArtistEmail

getArtistPhone

removeArtist

selectArtist

searchArtist

getName

validateGender

formaEmail

validateEmail

removeArtistAllAlbums

validatePhone

formatName

validateName

albumViewer

albumMenu

albumEditor

editAlbum

addAlbum

deleteAlbum

ViewArtistAlbumsBySearch

editAlbumMenu

viewAlbumMenu

displayAllAlbum

sortAlbum

selectAlbum

editAlbum

selectArtist

displayOneAlbum

searchAlbumByArtistId

displayAlbumSearchResult

searchArtist

displaSearchAlbumResult

SearchArtist

selectArtist

SearchAlbumByArtistId

SearchArtistNameById

searchArtist

selectArtist

selectAlbum

removeAlbum

getAlbumInfo

removeArtistAllAlbums

SearchAlbumByArtistId

selectArtist

SearchArtist

SortAlbum

getAlbumDate

getAlbumFormat

getAlbumTitle

formatAlbumPath

validateAlbumPath

formAlbumDate

formatAlbumDate

formatAlbumFormat

validateAlbumFormat

FormatAlbumTitle

getAlbumPath

validateAlbumTitle

**Data Structure**

**For Artist**

Two dimensional character array to stores artist id(array of c-style string): artistIds[1000][8]

A character array to store the gender(c-style string): genders[1000]

Two dimensional character array to stores artist names(array of c-style string): names[1000][40]

Two Dimensional character array to store phones(array of c-style string): phones[1000][11]

Two Dimensional character array to store emails(array of c-style string):: emails[1000][60]

**For Album**

Two dimensional character array to stores artist id reference(array of c-style string): artistIdRefs[1000][8]

Two dimensional character array to album ids(array of c-style string): albumIds[1000][10]

Two dimensional character array to stores Titles(array of c-style string): titles[1000][80]

Two Dimensional character array to store formats(array of c-style string): recordFormats[1000][12]

Two Dimensional character array to store published dates(array of c-style string): datePublisheds[1000][11]

Two Dimensional character array to store album paths(array of c-style string): paths[1000][100]

**Global data**

* lastArtistId(int) is the last added artist id(the largest)
* lastAlbumId(int) is the last added album id(the largest)

**Module description and interface definition**

1. **main**

The main function

**int main()**

Parameters

* No parameter

Return

1. **welcome()**

Displays a welcome message to the user

**void welcome()**

Parameters

* No parameter

Return

* No return

1. **loading**

Populate the artist and album arrays. ***For version one the loading will be done from a fixed array***

**void loading(char artistIds[][8], char names[][40], char genders[], char phones[][11], char emails[][80], char artistIdRefs[][8], char albumIds[][8], char titles[][80], char recordFormats[][20], char datePublished[][11], char paths[][100],int & nArtist, int & nAlbum)**

Parameters

* artistIds, names, genders, phones, emails are arrays that is to be loaded with artist information from globally declared arrays
* artistIdRefs, char albumIds, titles, recordFormat, datePublished, paths are arrays to be loaded with album information from globally declared arrays
* nArtist is the total number of artist records(to be set in this function)
* nAlbum is the total number of album records(to be set in this function)

Return

* The function doesn’t return.

1. **openFile(**Not needed for this version)

Open the artist and album file.

**bool openFile(fstream & fstrm)**

Parameters

* fstrm: a file stream to be associated with opened file(fstream type)

Return

* The function returns a status representing whether or not the file opening is successful

1. **loadArtist**

Populate the artist arrays. ***For version one the loading will be done from a fixed array***

**void loading(char artistIds[][8], char names[][40], char genders[], char phones[][11], char emails[][80], int & nArtist)**

Parameters

* artistIds, names, genders, phones, emails are arrays that is to be loaded with artist information
* nArtist is the total number of artist records loaded

Return

* The function doesn’t return.

1. **loadArtist**

Populate the album arrays. ***For version one the loading will be done from a fixed array***

**void loading(char artistIdRefs[][8], char albumIds[][8], char titles[][80], char recordFormats[][20], char datePublished[][11], char paths[][100], int & nAlbum)**

Parameters

* artistIdRefs, char albumIds, titles, recordFormat, datePublished, paths are arrays to be loaded with album information
* nAlbum is the total number of album records loaded

Return

* The function doesn’t return.

1. **sortArtist**

Sorts the artist information array by name.

**void sort(char artistIds[][8], char names[][40], char genders[], char phones[][11], char emails[][80], int nArtist)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays

Return

* The function returns nothing

1. **sortAlbum**

Sorts the album information array by artistIdsRefs

**void sort(char artistIdRefs[][8], char albumIds[][8], char titles[][80], char recordFormats[][20], char datePublished[][11], char paths[][100], int nAlbum)**

Parameters

* artistIdRefs, char albumIds, titles, recordFormat, datePublished, paths are arrays to be loaded with album information
* nAlbum is the number of albums information currently existing in the arrays

Return

* The function returns nothing

1. **mainHandler**

Handles the main menu choice

void main**(char artistIds[][8], char names[][40], genders[], char phones[][11], char emails[][80], char artistIdsRefs[][8], char albumIds[][8], char titles[][80], char recordFormats[][20], char datePublished[][11], char paths[][100], int & nArtist, int & nAlbum)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays
* artistIdRefs, char albumIds, titles, recordFormat, datePublished, paths are arrays to be loaded with album information
* nAlbum is the number of albums information currently existing in the arrays

Return

* The function returns nothing

1. **mainMenu**

Displays the main menu: (Manage Artist, Manage Album, Exit) and allow the user to choose and returns the user choice

**int mainMenu()**

Parameter

* No parameters

Return

* The function returns an integer value representing the user selection

1. **artistManager**

Manages artist information

**bool artistManager(char artistIds[][8], char names[][40], genders[], char phones[][11], char emails[][80], char artistIdsRefs[][8], char albumIds[][8], char titles[][80], char recordFormats[][20], char datePublished[][11], char paths[][100], int & nArtist, int & nAlbum)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays
* artistIdRefs, char albumIds, titles, recordFormat, datePublished, paths are arrays to be loaded with album information
* nAlbum is the number of albums information currently existing in the arrays

Return

* The function returns a bool value representing whether the user chooses exit or not

1. **ArtistMenu**

Displays the artist menu: (View Artist, Edit Artist, Back To Main Menu, Exit) and allow the user to choose and returns the user choice

**int artistMenu()**

Parameter

* No parameters

Return

* The function returns an integer value representing the user selection

1. **artistViewer**

Handles the view artist choice

**bool artistViewer(const char artistIds[][8], const char names[][40], const char genders[], const char phones[][11], const char emails[][80], int nArtist)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays

Return

* The function returns a bool value representing whether the user chooses exit or not

1. **viewArtistMenu**

Displays the views menu: (Display All Artist, View Artist By Search, Back To Main Menu, Exit), and allow the user to choose and returns the user choice

**int viewArtistMenu()**

Parameters

* No parameters

Return

* The function returns an integer value representing the user selection

1. **displayAllArtist**

Displays all artist with their detail

**void displayAllArtist(const char artistIds[][8], const names[][40], const genders[], const char phones[][11], const char emails[][80], int nArtist)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays

Return

* The function returns nothing

1. **viewArtistBySearch**

Allow the user to see artists by search

**void viewArtistBySearch(const char artistIds[][8], const names[][40], const genders[], const char phones[][11], const char emails[][80], int nArtist)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays

Return

* The function returns nothing

1. **searchArtist**

Searches Artist by name or id .The search is prefix search (start with). Returns the result through the result array which contains the index positions If target starts with digit search will be by id otherwise by name

**void searchArtist(const char artistIds[][8], const names[][40], int nArtist, int result[], int \* noResult)**

Parameters

* artistIds, names are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays
* result is an array that is going to contain the index position of the matching records
* noResult is the number of matching record

Return

* The function returns nothing

1. **searchArtistById**

Searches artist by id .The search is prefix search (start with)

**void searchArtistById(const char artistIds[][8], int nArtist, const char targetId[], int result[], int \* noResult)**

Parameters

* artistIds, is an array that contains the artist id in an array
* nArtist is the number of artist information currently existing in the arrays
* result is an array that is going to contain the index position of the matching records
* target is the target id to be searched
* noResult is the number of matching record

Return

* The function returns nothing

1. **searchArtistByName**

Searches artist by id .The search is prefix search (start with)

**void searchArtistByName(const names[][40], int nArtist, const char targetName[], int result[], int \* noResult)**

Parameters

* names, is arrays that contains the artist names information in an array
* nArtist is the number of artist information currently existing in the arrays
* result is an array that is going to contain the index position of the matching records
* targetName is the target name to be searched
* noResult is the number of matching record

Return

* The function returns nothing

1. **displaySearchResult**

Displays search result

**void displaySearchResult(const char artistIds[][8], const names[][40], const genders[], const char phones[][11], const char emails[][80], int nArtist, const int result[], int noResult)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays
* result is the search result array
* noResult is the number of matching record

Return

* The function returns nothing

1. **artistEditor**

Handles the edit artist choice. Allow the user to add, edit or remove artist

**bool artistEditor(char artistIds[][8], char names[][40], char genders[], char phones[][11], char emails[][80], char artistIdsRefs[][8], char albumIds[][8], char titles[][80], char recordFormats[][20], char datePublished[][11], char paths[][100], int & nArtist, int & nAlbum)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays
* artistIdRefs, char albumIds, titles, recordFormat, datePublished, paths are arrays to be loaded with album information
* nAlbum is the number of albums information currently existing in the arrays

Return

* The function returns a bool value representing whether the user chooses exit or not

1. **editArtistMenu**

Displays the edit artist menu: (Edit Artist, Delete Artist, Back To Main Menu, Exit), and allow the user to choose and returns the user choice

**int viewArtistMenu()**

Parameters

* No parameters

Return

* The function returns an integer value representing the user selection

1. **addArtist**

Add artist and return status

**bool addArtist(char artistIds[][8], char names[][40], char genders[], char phones[][11], char emails[][80], int & nArtist)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays

Return

* The function returns bool representing success or failure

1. **getArtistInfo**

This gets artist info from user

**void getArtistInfo (char name[], char & gender, char phones[], char emails[])**

Parameters

* name, gender, phone, email are variables to hold artist info

Return

* The function returns nothing

1. **getArtistName**

This gets artist name from user

**void getArtistName (char name[])**

Parameters

* name, is artist full name

Return

* The function returns nothing

1. **getArtistGender**

This gets artist gender from user

**char getArtistGender ()**

Parameters

* No parameter

Return

* The function returns char representing the artist gender

1. **getArtistPhone**

This gets artist phone from user

**void getArtistPhone (char phone[])**

Parameters

* phone, is artist phone

Return

* The function returns nothing

1. **getArtistEmail**

This gets artist email from user

**void getArtistEmail (char email[])**

Parameters

* email, is artist email

Return

* The function returns nothing

1. **validateName**

This validates artist name

**bool validateName (const char name[])**

Parameters

* name, is artist full name

Return

* The function returns true if the name is valid otherwise returns false

1. **FormatName**

This formats artist name

**void formatName (char name[])**

Parameters

* name, is artist full name

Return

* No return

1. **validateGender**

This validates artist gender

**bool validateGender( char gender)**

Parameters

* gender, is artist gender

Return

* The function returns true if the gender is valid otherwise returns false

1. **validatePhone**

This validates artist phone

**bool validatePkone (const char phone[])**

Parameters

* phone, is artist phone

Return

* The function returns true if the phone is valid otherwise returns false

1. **validateEmail**

This validates artist email

**bool validateEmail (const char email[])**

Parameters

* email, is artist email

Return

* The function returns true if the name is valid otherwise returns false

1. **FormatEmail**

This formats artist email

**void formatEmail (char email[])**

Parameters

* email, is artist email

Return

* No return

1. **editArtist**

allow the user to edit artist(handles edit artist menu item choice)

**void editArtist(const char artistIds[][8], char names[][40], char genders[], char phones[][11], char emails[][80], int nArtist)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays

Return

* No return value

1. **selectArtist**

Display the search result and allow the user to select the artist to edit. And returns the selected artist

**int selectArtist(const char artistIds[][8], const char names[][40], const int result[], int noResult. int forWhat )**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* result is search result array containing the index of the matching artist
* noResuit is the number of matching artist in result array
* forWhat 0 for view,1 for edit 2, for delete

Return

* Returns the selected artist position

1. **editArtistInfo**

Allow the user to edit field by field until the user is done. Returns status(changed or not)

**bool editArtistInfo(const char artistId, char name[], char gender, char phone[],char email[])**

Parameters

* name, phone, email are strings and gender is a char that contains the artist information to be edited

Return

* Returns true if the record is changed otherwise false

1. **displayOneArtist**

Display a single artist vertically

**void displayOneArtist(const char artistId[], const char name[], char gender, const char phone[], const char email[])**

Parameters

* artistId, name, phone, email are strings and gender is a char that contains the artist information to be displayed

Return

* No return

1. **deleteArtist**

Handles delete choice.

**void deleteArtist(char artistIds[][8], char names[][40], char genders[], char phones[][11], char emails[][80], char artistIdsRefs[][8], char albumIds[][8], char titles[][80], char recordFormats[][20], char datePublished[][11], char paths[][100], int & nArtist, int & nAlbum)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays
* artistIdRefs, char albumIds, titles, recordFormat, datePublished, paths are arrays to be loaded with album information
* nAlbum is the number of albums information currently existing in the arrays

Return

* No return value

1. **removeArtist**

Handles delete choice

**void removeArtist(char artistId[][8], char name[][40], char phone[][11],char email[][80], int selectedIdx, int & nArtist)**

Parameters

* artistIds, names, genders, phones, emails are arrays that contains the artist information in an array
* nArtist is the number of artist information currently existing in the arrays
* selectedIndex is the index of the target to be deleted

***Note: to delete an artist first all albums of the artist must be deleted***

Return

* No return value

1. **removArtistAllAlbums**

removes all albums of the artist

**void removeArtistAllAlbums(const char artistId[], char artistIdsRefs[][8], char albumIds[][8], char titles[][80], char recordFormats[][20], char datePublished[][11], char paths[][100], int & nAlbum)**

Parameters

* artistId is the id of the artist whose albums to remove
* artistIdRefs, char albumIds, titles, recordFormat, datePublished, paths are arrays to be loaded with album information
* nAlbum is the number of albums information currently existing in the arrays

Return

* No return value

1. **albumManager**

Manages album information

**bool albumManager((const char artistIds[][8], const char artistNames[][40], int nArtist, char artistIdsRef[][8], char albumIds[][10], char titles[][80], char recordFormats[][12], char datePublisheds[][11], char paths[][100], int & nAlbum)**

Parameters

* artistIds, artistNames are array containing artist id and names from artist information
* nArtist is the number of record in artist arrays
* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the artist information in an array
* nAlbum is the number of album information currently existing in the arrays
* lastAlbumId is the last added added id(the largest)

Return

* The function returns a bool value representing whether the user chooses exit or not

1. **AlbumMenu**

Displays the album menu: (View Artist Albums, Edit Artist Album, Back To Main Menu, Exit) and allow the user to choose and returns the user choice

**int albumMenu()**

Parameter

* No parameters

Return

* The function returns an integer value representing the user selection

1. **albumViewer**

Handles the view album choice

**bool albumViewer(const char artistIds[][8], const char artistNames[][40] ], int nArtist, const char artistIdsRef[][8], const char albumIds[][10], const char titles[][80], const char recordFormats[][12], const char datePublisheds[][11], const char paths[][100], int nAlbum)**

Parameters

* artistIds, artistNames are array containing artist id and names from artist information
* nArtist is the number of record in artist arrays
* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the album information in an array
* nAlbum is the number of album information currently existing in the arrays

Return

* The function returns a bool value representing whether the user chooses exit or not

1. **viewAlbumMenu**

Displays the views menu: (Display All Album, View Album By Search, Back To Main Menu, Exit), and allow the user to chose and returns the user choice

**int viewAlbumMenu()**

Parameters

* No parameters

Return

* The function returns an integer value representing the user selection

1. **displayAllAlbums**

Displays all album with their detail categorized by artistId

**void displayAllAlbums(const char artistIds[8], const char artistNames[40], const char artistIdsRef[][8], const char albumIds[][10], const char titles[][80], const char recordFormats[][12], const char datePublisheds[][11], const char paths[][100], int nAlbum)**

Parameters

* artistIds, artistNames are the id and name of the artist whose albums we want to display
* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the album information in an array
* nAlbum is the number of album information currently existing in the arrays

Return

* The function returns nothing

1. **searchAlbumByArtistId**

Searches artist by artist id. The result contains all album of a given artist

**void searchAlbumByArtistId(const char artistIdsRefs[][8], int nAlbum, const char target[], int result[], int \* noResult)**

Parameters

* artistIdsRefs, is an array that contains the artist id reference in album information
* nAlbum is the number of albums information currently existing in the arrays
* result is an array that is going to contain the index position of the matching records
* target is the target artist id to be searched
* noResult is the number of matching record

Return

* The function returns nothing

1. **displayAlbumSearchResult**

Displays album search result

**void displayAlbumSearchResult(const char artistIds[8], const char artistNames[40], const char artistIdsRef[][8], const char albumIds[][10], const char titles[][80], const char recordFormats[][12], const char datePublisheds[][11], const char paths[][100], const int result[], int noResult)**

Parameters

* artistIds, artistNames are the id and name of the artist whose albums we want to display
* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the album information in an array
* nAlbum is the number of album information currently existing in the arrays
* nArtist is the number of artist information currently existing in the arrays
* result is the search result array
* noResult is the number of matching record

Return

* The function returns nothing

1. **albumEditor**

Handles the edit album choice. Allow the user to add, edit or remove album

**bool albumEditor(const char artistIds[][8], const char artistNames[][40], int nArtist, char artistIdsRef[][8], char albumIds[][10], char titles[][80], char recordFormats[][12], char datePublisheds[][11], char paths[][100], int & nAlbum)**

Parameters

* nArtist is the number of artist information currently existing in the artist arrays
* artistIds, artistNames are array containing artist id and names from artist information
* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the album information in an array
* nAlbum is the number of album information currently existing in the arrays

Return

* The function returns a bool value representing whether the user chooses exit or not

1. **editAlbumMenu**

Displays the views menu: (Add Artist Album, Edit Artist Album, Delete Artist Album, Back To Main Menu, Exit), and allow the user to choose and returns the user choice

**int viewArtistMenu()**

Parameters

* No parameters

Return

* The function returns an integer value representing the user selection

1. **addAlbum**

handles the add artist album menu item

**bool addAlbum(const char artistIds[][8], const char artistNames[][40], int nArtist, artistIdsRef[][8], char albumIds[][10], char titles[][80], char recordFormats[][12], char datePublisheds[][11], char paths[][100], int & nAlbum)**

Parameters

* artistIds, artistNames are array containing artist id and names from artist information
* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the album information in an array
* nAlbum is the number of album information currently existing in the arrays

Return

* The function returns bool representing success or failure

1. **getAlbumInfo**

This gets album info from user

**void getArtistInfo (char titles[], recordFormats[], char datePublisheds[], char paths[])**

Parameters

* titles, recordFormat, datePublished, path are variables to hold album info

Return

* The function returns nothing

1. **getAlbumTitle**

This gets album title from user

**void getAlbumTitle (char title[])**

Parameters

* title, is album title

Return

* The function returns nothing

1. **getAlbumRecordFormat**

This gets album record format from user

**void getAlbumRecordFormat(char albumFormat[])**

Parameters

* albumFormat is a string representing the format of the record to be filled

Return

* No return

1. **getAlbumDate**

This gets the date on which the album is published

**void getAlbumDate (char albumDate[])**

Parameters

* albumDate is a string representing the date of the album published to be filled

Return

* The function returns nothing

1. **getAlbumPath**

This gets the directory path of the album from user

**void getAlbumPath (char albumPath[])**

Parameters

* albumPath is a string representing the album path of the album

Return

* The function returns nothing

1. **validateAlbumTitle**

This validates Album title

**bool validateAlbumTitle (const char albumTitle[])**

Parameters

* title, is album title

Return

* The function returns true if the title is valid otherwise returns false

1. **FormatAlbumTitle**

This formats album title

**void formatAlbumTitle (char albumTitle[])**

Parameters

* title, is album title

Return

* No return

1. **validateAlbumFormat**

This validates Album

**bool validateAlbumFormat (const char albumFormat[])**

Parameters

* albumFormat, is album record format

Return

* The function returns true if the format is valid otherwise returns false

1. **formatAlbumFormat**

This formats album record format

**void formatAlbumFormat(char albumFormat[])**

Parameters

* albumFormat, is album record format

Return

* No return

1. **validateAlbumDate**

This validates album path

**bool validateAlbumDate(unsigned int day, unsigned int mmonth, unsigned int year)**

Parameters

* day, month, year is the date the album is published

Return

* The function returns true if the path is valid otherwise returns false

1. **FormatAlbumDate**

This formats album date

**void formatAlbumPath(char albumDate[], unsigned int day, unsigned int month, unsigned int year)**

Parameters

* day, month, year is the date the album is published
* albumPath, is used to return the formatted date of the album

Return

* No return

1. **validateAlbumPath**

This validates album path

**bool validateAlbumPath(const char albumPath[])**

Parameters

* albumPath, is path of the album

Return

* The function returns true if the path is valid otherwise returns false

1. **FormatAlbumPath**

This formats album path

**void formatAlbumPath(char albumFormat[])**

Parameters

* albumPath, is the path of the album

Return

* No return

1. **editAlbum**

handles the Edit Artist Album menu item

**void editAlbum(const char artistIds[][8], const char artistNames[][40], int nArtist, char artistIdsRef[][8], char albumIds[][10], char titles[][80], char recordFormats[][12], char datePublisheds[][11], char paths[][100], int nAlbum)**

Parameters

* nArtist is the number of artist information currently existing in the artist arrays
* artistIds, artistNames are array containing artist id and names from artist information
* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the album information in an array
* nAlbum is the number of album information currently existing in the arrays

Return

* No return value

1. **selectAlbum**

Display the search result and allow the user to select the album to edit or remove. And returns the selected album

**void selectAlbum(const char artistId[], const char artistName[], const char artistIdsRef[][8], const char albumIds[][10], const char titles[][80], const char recordFormats[][12], const char datePublisheds[][11], const char paths[][100], const int result[], int noResult. int forWhat )**

Parameters

* ArtistId and artistName are the id and name of the artist whose album is going to be edited
* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the album information in an array
* result is search result array containing the index of the matching artist
* noResuit is the number of matching artist in result array
* forWhat o for view,1 for edit 2 for delete

Return

* Returns the selected album position

1. **editAlbumInfo**

Allow the user to edit Album field by field until the user is done. Returns status(changed or not)

**bool editAlbumInfo(const char artistIdsRef[], const char albumIds[], char titles[], char recordFormats[], char datePublisheds[], char paths[])**

Parameters

* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the album information to be edited

Return

* Returns true if the record is changed otherwise false

1. **displayOneAlbum**

Display a single album vertically

**void displayOneArtist(const char artistIdsRef[], const char albumIds[], char titles[], char recordFormats[], char datePublisheds[], const char paths[])**

Parameters

* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the album information to be edited

Return

* No return

1. **deleteAlbum**

Handles delete choice.

**void deleteAlbum(const char artistIds[][8], const char artistNames[][40], int nArtist, char artistIdsRef[][8], char albumIds[][10], titles[][80], recordFormats[][12], char datePublisheds[][11], char paths[][100], int & nAlbum**)

Parameters

* nArtist is the number of artist information currently existing in the artist arrays
* artistIds, artistNames are array containing artist id and names from artist information
* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the album information in an array
* nAlbum is the number of album information currently existing in the arrays

Return

* No return value

1. **removeAlbum**

removes album

**void removeAlbum(char artistIdsRef[][8], char albumIds[][10], titles[][80], recordFormats[][12], char datePublisheds[][11], char paths[][100], int & nAlbum, int selectedIdx,)**

Parameters

* artistIdsRef, albumIds, titles, recordFormats, datePublisheds, paths, are arrays that contains the album information in an array
* nArtist is the number of artist information currently existing in the arrays
* selectedIndex is the index of the target to be deleted

Return

* No return value

1. **farewell()**

Displays a farewell message to the user

**void farewell ()**

Parameters

* No parameter

Return

* No return