Introduction

Due to its great success, you have been asked to extend your music collection program. Customers like what they see but want more! In particular they want you to add the following capabilities. First, it should be possible to create new collections and then save them to a file. It should also be possible to open a collection by entering the appropriate file name. The ability to search the current collection for a particular phrase was also a frequently requested new option. Finally, it is suggested that it be possible to refer to a song in a collection by a *track number* — a numeric code which indicates the song's position in the collection.

New Commands

These new requests will be provided by implementing the following new menu options — the associated capability is also described.

open Asks the user for the name of a collection file, opens the file and reads the current songs from the file into the `Collection` object. If the user enters an invalid file name (one that can't be opened), print an error message but don't ask for another name. It is assumed that the user will choose this option again if she wants to correct the file name.

If the current collection contains some songs then the user should be prompted to select either the "close" or "save" option before selecting "open" again.

close The current collection is removed by having the `Collection` object send the `resize(0)` message to the `vector` object and set its size to zero.

As this option has the possibility to destroy data, the user should be asked for confirmation before actually deleting the current collection.

new Allows the user to start a new song collection.

If the current collection contains some songs then the user should be prompted to select either the "close" or "save" option before selecting "open" again.

save Saves the current collection to the file associated with the collection. If the collection hasn't been assigned a file name yet (i.e., it was just created) then prompt for the name and open the file for writing.

Be sure to validate the file name with a validation loop to ensure that it can be opened.

track The user will be prompted for an integer value between 1 and the collection's maximum track number; the program will list the song information for the song with the specified track number.

find The user types in a target string and the program will display the information for the first song found in the collection that contains the target string in either the title or artist portion.

The program should also allow the user to only type in the first letter of each command as a short cut.

New Structures

In order for the new options to work you will need to add a new data member to each of the classes `Collection` and `Song`. To the `Collection` class you will add a string data member which will hold the name of the file the collection is stored in. In this way the program can know where the collection should be written to when the `save` command is selected. In addition, you will add a new integer data member to the `Song` class which holds the track number for the song. The track number is one more than the song's position in the `Collection` vector.

Structure of the Data File

The format of of a collection file is as a text file, with each line containing the information for one song. The information for a song has the following form: the song title, artist and minutes and seconds separated by ":" and a ":" at the end. For example, here is a sample file with two songs:

```
Oops, I Did It Again!:Britney Spears:3:13:
Are You Ready?:Creed:12:45:
```

We suggest you use `getline(inputStream, part, ':');` to read each part of the file where `part` is a string. You can convert the string "12" to the `int` 12 by using the function `atoi(part.c_str())`. You can once again use the `getline()` function from `ifstream`, just as you did in the last project.

You may assume that collection files are in the correct format.

What To Hand In

Hand in a copy of your program and test runs that convince us that all the commands and features work. Remember to use good style and to write Pre and Post conditions for all methods.