Introduction

You are to write a C++ program to organize your MP3 song titles. Each song will have a song title, artist name and length in minutes and seconds. The song list will be stored in a data file. The user interacts with the program by issuing text commands. The commands and their actions are listed below.

User Commands

The input of the program will be a text-oriented user interface where the user types in one of the following commands after the prompt:

```
Command:
```

add Allow the user to enter the information of a new song and add to the end of the `collection` vector.

help Types out a brief description of all of the commands.

list List the songs in the `collection` vector.

quit Quits the program.

total Prints out the total time in minutes and seconds of all the songs in the `collection` vector.

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

Structure of the Program

The main program should create a `Collection` object which consists of a vector of `Song` objects. The main program should repeatedly ask the user to type in a command until the user types in the `quit` command. For each command (except `quit`), your program should call a corresponding method in the `Collection` class which will perform the actions associated with that command.

The `Song` objects hold the song title, artist name and length in minutes and seconds.

You will need to write the specification and implementation files for the `Collection` class and the `Song` class.

Structure of the Data File

The format of a line in the data file is 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(stream, 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())`.

When using the above `getline()` functions, you will need to read the '`\n`' at the end of the line. One way is to use

```
getline(inFile, dummy, '\n');
```

You may assume the file is in the correct form.

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.