Playlist Project **AP Computer Science**

Due:

For this project, you and up to two partners (if you like!) will work on creating a series of files designed to mimic a playlist of songs much like a Spotify or iTunes playlist. As part of practicing with GitHub, your submission will be a **link to your GitHub repository**. To receive full credit, your project must have a history of **at least 5 commits with included messages.** All partners must submit at least one commit.

A series of "starter" files is available for you to use—if you would like to approach the project differently, that is up to you!

To make life easier, your tester program should **not** include any opportunity for user input—instead all of your tester implementation will be hard-coded and printed out to show viability. This means that you should **not make a Scanner** for your project.

The `Playlist` object should contain a collection of `Song` objects—each `Song` should have a title, artist, duration and status of whether the song is "liked" associated with it.

The `Playlist` should demonstrate the following functionality:

- adding a song to the playlist
- "liking" a song in a playlist
- removing a song from a playlist
- examining all songs in the playlist
- examining a sublist of just the "liked" songs
- determining the total duration of the playlist
- removing *all* "unliked" songs in a playlist with a single method—be careful when removing elements from a list (what is a common bug with removing from a list?)

The `PlaylistTester` file should include a demonstration of each of these methods that the `Playlist` has. On the back page, you can see sample output of what your finished code should produce when running.

Description	**Points Possible**
The `Song` class is written with standard object-oriented conventions and is used accurately in the rest of the program. All necessary methods and fields are provided in the program.	5
The `Playlist` class is written with standard object-oriented conventions and is used accurately in the rest of the program. All necessary methods and fields are provided in the program.	10
The `PlaylistTester` class creates a `Playlist` and demonstrates all necessary functionality of the given `Playlist` in a logical and thoughtful manner.	10
The project is successfully uploaded to GitHub with a history of at least 5 commits, each of which contain a commit message. All group members must have submitted at least one commit of their own.	15

Sample output:

```
Initializing a Playlist...

Adding songs to the Playlist...

Added “This Must Be the Place” by Talking Heads (4:56)
Added “Dominoes” by Donald Byrd (4:33)
Added “Check the Rhime” by A Tribe Called Quest (3:36)
Added “Con Altura” by Rosalia (2:41)
Added “California” by Joni Mitchell (3:50)


Printing the songs...

“This Must Be the Place” by Talking Heads (4:56)
“Dominoes” by Donald Byrd (4:33)
“Check the Rhime” by A Trible Called Quest (3:36)
“Con Altura” by Rosalia (2:41)
“California” by Joni Mitchell (3:50)



Liking the songs in position 1, 3 and 4...

Printing the songs...

“This Must Be the Place” by Talking Heads (4:56)
“Dominoes” by Donald Byrd (4:33) -- liked
“Check the Rhime” by A Trible Called Quest (3:36)
“Con Altura” by Rosalia (2:41) -- liked
“California” by Joni Mitchell (3:50) – liked



Removing the song in position 2...

Printing the songs...

“This Must Be the Place” by Talking Heads (4:56)
“Check the Rhime” by A Trible Called Quest (3:36)
“Con Altura” by Rosalia (2:41) -- liked
“California” by Joni Mitchell (3:50) – liked


Printing only the liked songs...

“Con Altura” by Rosalia (2:41) -- liked
“California” by Joni Mitchell (3:50) – liked


Printing the total duration of all songs...
15:03


Removing all unliked songs at once...

Printing all songs...

“Con Altura” by Rosalia (2:41) -- liked
“California” by Joni Mitchell (3:50) – liked
```