File Reading Lab with the Billboard Hot 100

The Billboard Hot 100 is a record chart indicating the popularity of songs in the United States. The list was first published in the 1950s, with the birth of rock and roll, and has been updated every week since then. Song popularity is currently determined by song sales, radio play, and online streaming. The list is presented in ranked order, so the most popular song for the week is first on the list, the second-most popular song is next, and so on.

You are given a file `songs.txt` that contains information about the top 20 songs on the Billboard Hot 100 chart. The songs in the file are presented in the same order as the Billboard list, meaning the first line of the file contains the #1 song for the week (the most popular), the second line has the #2 song (second most popular), and so on.

Each line of the file contains information about one song:

- The song's title
- The song's artist
- The song's ranking on the chart during the *previous* week (`-1` if this song was not on the chart during the previous week)
- The number of weeks the song has been on the chart (in any position on the chart) (e.g., `1` if this song was not on the chart during the previous week)

A semicolon (`;`) is used to separate these four parts on a single line of the file.

Write the following programs:

1. Write a program to open the file and print all the information in the file. Make sure to correctly open the file, use a for loop to read each line, `rstrip()` the line, and split the line into the four pieces. Don't forget to convert the ranking on the chart and the number of weeks on the chart into integers!

2. Write a program to print the names and artists of all the songs that have been on the chart for at least 20 weeks.

3. Write a program to print the name and artist of the song that has been on the chart the longest (greatest number of weeks), and the name and artist of the song that has been on the chart the fewest number of weeks. (If there are songs that are tied for greatest or fewest number of weeks, you may choose any one you like to print.)

   Hint: first write the program to just find the largest and smallest number of weeks on the chart, then change your code so it also saves the artist and title, as well as the number of weeks.

4. Write a program to print the name of each song on the chart, along with a message saying whether
   - the song moved *up* the chart from the previous week (that is, its ranking on the chart this week is less than its ranking the previous week), and how much the ranking changed,
   - the song moved *down* the chart from the previous week, and how much the ranking changed,
   - the song *stayed the same* when compared to the previous week, or
   - the song is a *new song* on the chart (it wasn't even on the chart the previous week).

   Obviously only one of the four situations will apply to each song on the chart.

   Hint: The file directly gives you each's song's ranking on the chart for the *prior* week, but not the *current* week. However, you can compute the ranking in the current week very easily, given what you know about how the file is organized.

5. Write a program to find all pairs of consecutively-ranked songs on the chart where their relative popularities are reversed from the prior week. In other words, find all back-to-back songs in the file where --- in the prior week --- the currently less-popular song was ranked higher than the currently more-popular song. Note that the songs need not be consecutively-ranked in the prior week, just the current week. Hint: Use the sliding window technique.