CSCI-14 Final programming project -- using arrays of structs and file I/O. 100 points.
Due 11:59:00 PM the last day of classes before the start of final exam week. Late (-20%) before the start of the final. No projects accepted at all after the start of the final. No other work accepted after the last class day.

Write a program to help a baseball team computerize its calculation of players batting and on base percentages. There are 20 players on the team, listed in a file in the format "last-name first-name" (separated by a space, with a maximum of 25 characters each) for each player, as follows:

```
Smith Sam
Jones Joe
Miller Miles
```

and so forth for 20 players. The players will be numbered 1-20 in the order listed in this file. Notice that there are no numbers in this file. You may assume the players will have unique last names and that there will be exactly 20 players in the file. You may assume the player file will be formatted correctly.

A separate file contains the game-by-game statistics for the players as a "player number, hits, walks, outs" (separated by comma ',' and a tab or space) for each game as follows:

```
3, 2, 1, 1     // e.g. player 3 got 2 hits, 1 walk and 1 out in this game (in 4 at-bats)
5, 1, 1, 2
3, 1, 2, 1
```

and so forth for the season (perhaps many games). Notice that the same player number could appear several times, or that a player might not play in a game at all during the season. You may assume the file is also formatted correctly. The last line in this file will contain `-1, -1, -1, -1`. You may not assume any particular limit to the length of this file (the report could be run before the first game was played, there could be no season because of a strike, or after a season of many games the team could go into post-season play). A player number in this file might not exist in the roster. In this case, print a message to the output file to say something like “Player 25 is not on the roster”.

Create an array of PLAYER structs to hold the player data. Read the players into the array then sort them by last name. You must keep the original player numbers associated with the players' names after the sort. Then, read the statistics file, updating each player's statistics for number of games played and number of hits, walks and outs. After reading all of the statistics, calculate batting average ((hits) / (hits + outs)) (this does not include walks at all), and on base average ((hits + walks) / (total at bats)). Keep these values in the PLAYER struct also.

Print the following to the output file, in this order:

- the input and output file names for this run
- the alphabetical list of players with each player's statistics (omit inactive players)
- a roster in sorted order by batting average, highest to lowest of all active players (omit inactive players!)
- a roster in sorted order, by on-base percentage, highest to lowest of all active players (omit inactive players)
- the name and batting average of the best hitter
- the name and on-base percentage of the best base runner
- the team hitting and on-base averages

- the name and batting average of the worst hitter (omit inactive players)
- the name and on-base percentage of the worst base runner (omit inactive players)
- a roster for the team in player number order with their numbers (all players, whether active or not)

Create (at least) 2 player files with 20 different players each in different orders and several "seasons" of data. For one of your tests, use my players and statistics file. You may use my players file for more than one season. You may use the same player file with several seasons of data for several program runs. Prompt for the file names and print the names of the input and output files at the top of your output. Send me your data and output files when you turn in the source file for your program. You do not need to send me my back own data files, just the output of the test using them. Include test runs where no one plays (i.e., the season is cancelled), where only some of the players play, and where all of the players play at least one game. Include at least one season of at least 20 games, with 9 (or more) players playing in each game (this will mean an input file of approximately 200 lines or more of player statistics). Note that you could not have some value I ask for in the output, i.e., if no games are played, there are no best or worst players, etc. If this happens, you need to tell me in output that there is no value.

Use a good modular design, with functions to do the separate tasks. <u>You may not use global variables at all in this program.</u>

One of the functions you will need to write is a search function to find a player record (by its index) searching on player ID. A player ID may be in error in the input file, so you have to handle that case, too.

You will also need a sort function, to sort a number of different ways; for example, by player number or by last name. This can actually be done in a single sort function. Work out how to do that.

Hints:

You need to keep the player's identification number (the key) with the player's names and the hit/walk/out data in the PLAYER struct.

When you read a player's game line, look up the player number in the array of structs.

You will need to track the number of games the player has played in, which is the number of lines in the statistics file that refer to the player. Since a player could have never played, you will need to check for "no games" happening and handle that in the calculation and output routines.