Consider the Number Guessing Game. In this game the computer chooses a random number between 1 and 100, and the player tries to guess the number in as few attempts as possible. Each time the player enters a guess, the computer tells him whether the guess is too high, too low, or right. Once the player guesses the number, the game is over.

For this assignment, you will be implementing a multiplayer version of this game. Your program should keep track of the best game (the game with the smallest number of guesses) for each player. The best guesses should be stored in an array. You can assume that there will never be more than 20 players. Initially, the entries of this array need to be initialized to -1 to indicate that no game has been played yet. When the program starts the user is presented with three options:

1. Play a new game. When this option is selected, your program should ask the user for their player id (a number between 1 and 20 to identify each player). The program will then generate a random number between 1 and 100 for the user to guess. The number of guesses is recorded and if it is less than the previously recorded best score, then the best score for this player is updated.

2. Show my best score. When this option is selected, your program should ask the user for their player id and print her best game score. If no game has been played by this user (best score is -1), your program should print a message letting the player know (instead of printing -1).

3. Show overall best score. When this option is selected, your program should print the best game score among all the players. If no game has been played (all scores are -1), your program should print a message stating that no games have been recorded yet.

In your second homework assignment (which will be implemented using the 3pi robots), you will work with your partner to implement this version of the number guessing game. To help you prepare for your future robot assignment, in this first homework assignment, you will **individually** implement this version of the game by writing a regular C/C++ program. By implementing this version of the game, you will have each individually been able to work out the logic of how the basic flow of the game should work in a C/C++ program. This will serve you well in completing the 3pi assignment with your partner for homework 2.

You have a lot of flexibility in how you will exactly implement this game for this individual homework assignment, but the following provides some general requirements:

- The players' best scores should be represented using an array with 20 elements.
- The best scores array should be initialized to -1.
- You need to implement a function that returns the best overall score by traversing an array and determining the best score. The main program should call this function when the user selects option 3 from the menu. The function prototype is provided in the starting files.
- Each time you start a new game, the number to be guessed should be different.
- You must implement your program in the file named hw1.cpp within your hw1 folder in your homework directory.
- You are allowed to use cin/cout or printf/scanf to interact with the console and gather input from the user.
- You are NOT allowed to use goto.
- In addition to having a standard documentation block and providing comments throughout your program, you must provide high-level pseudocode at the level of detail that someone with sufficient C knowledge would be able to implement your program. Pseudocode should be included in the documentation block of your program.

The following is a sample output for the game:

```
Welcome to the Multiplayer Number Guessing Game

1. Play a new game
2. Show my best score
3. Show overall best score
Please select a choice (or -1 to end): 3

No games recorded yet.

1. Play a new game
2. Show my best score
3. Show overall best score
Please select a choice (or -1 to end): 1

Enter your player id (1-20): 3
Enter a guess between 1 and 100: 50
Too low!
Enter a guess between 1 and 100: 90
Too high!
Enter a guess between 1 and 100: 75
Too high!
Enter a guess between 1 and 100: 72
Correct! You got it in 4 guesses!
Your best score = 4

1. Play a new game
2. Show my best score
3. Show overall best score
Please select a choice (or -1 to end): 2

Enter your player id (1-20): 3
Your best score = 4

1. Play a new game
2. Show my best score
3. Show overall best score
Please select a choice (or -1 to end): 3

Overall best score = 4

1. Play a new game
2. Show my best score
3. Show overall best score
Please select a choice (or -1 to end): -1

Thanks for playing! Bye bye.
```

General grading guidelines

Documentation and pseudocode (20 points)

- 10 points: including programmer, filename, and description aspects of top documentation block in-

cluding pseudocode (since you need to include pseudocode, the description can be brief)

- 10 points: Well documented code. This includes reasonable comments throughout the program (including having documentation blocks for any functions you may write yourself). Using good variable names, function names, and indentation style is an integral part of your documentation process.

Program (80 points)

- 5 points: correctly declaring an array to represent the best scores.
- 5 points: correctly initializing all elements of the array to -1.
- 15 points: correclty implementing the logic of the main menu.
- 10 points: correctly implementing the *getOverallBest* function.
- 15 points: correct logic for playing the guessing game.
- 10 points: correct update of the best scores array as the users play.
- 5 points: correclty calling the *getOverallBest* function when option 3 is selected.
- 5 points: correct output message when score is -1.
- 10 points: code organization (including not repeating a large amount of code) and ease of use of your program (e.g., having appropriate cues as to what the user should do while it is running)

Extra credit (up to 10 points)

To receive up to 10 extra credit points, you should add a fourth option to the menu to display the leader score board. This option should print the three best scores in order including the player id holding the records. In the case of ties all the player ids should be listed in the same category (first, second, or third place). Note that it is expected that you will work on the extra credit portion without help from the instructors or TA's. (You may ask clarification questions.) Extra credit is only possible if all other components of the program are working. Please do not attempt the extra-credit option until everything else is working.

Submission

The final version of your program must be committed to svn by Tuesday, February 9, 2016 at 11:59 p.m.