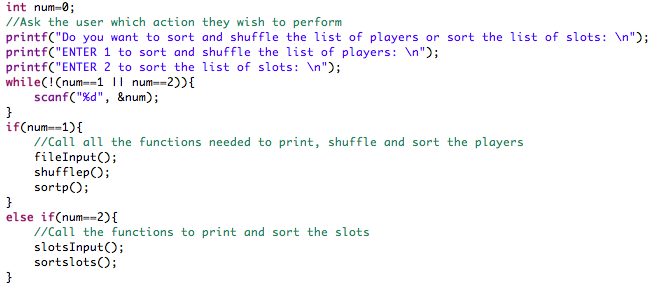
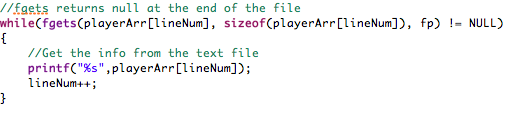
My program first takes input from the user as to whether to shuffle and sort the list of players or just to sort the list of slots. I used a while loop to ensure the user could only pick one of these two options, as well as ensuring they could not input an incorrect symbol.

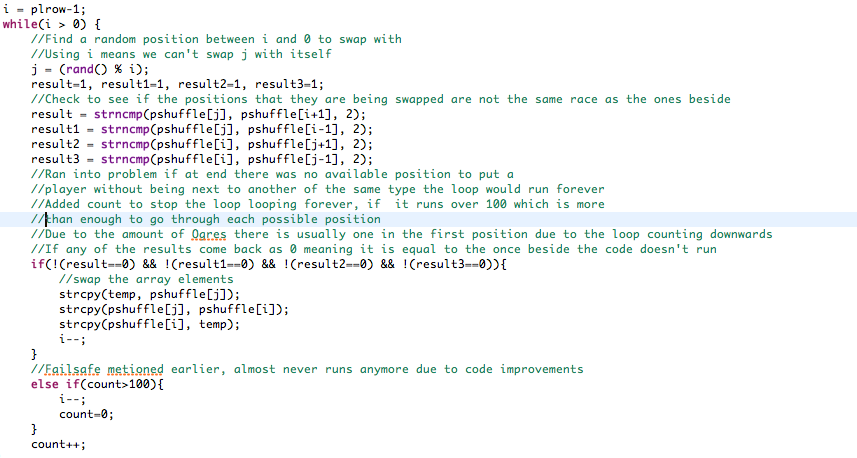


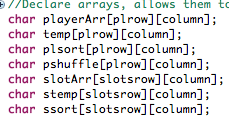
First I will be showing the operation of the player option. The first step was to input the players.txt file into the code so it could later be shuffled and sorted.

The following code took each line from the text file using fgets, and inserted it into an array, using this array I could copy it into different arrays which were later sorted and shuffled.

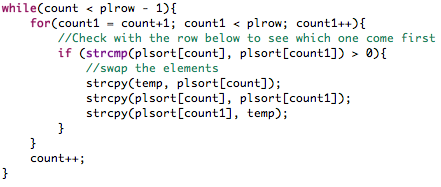
Scanf would not have been able to perform this action, leaving fgets as the clear option.

I ran into some issues when trying to shuffle the players while keeping players of the same race apart. It was relatively easy to set up the shuffling element first as was the suggestion in the question. With a different configuration, counting downwards would end up with the loop running infinitely as there were no possible places to swap the player with being beside the same race, this forced me to add a failsafe, using count to see if the loop ran over 100 times. Once this condition was satisfied I was decremented and count was reset. Unfortunately this led to a lot of cases where players of the same race ended up beside each other. The next idea was to increment I instead, in order to give j more positions to select and swap with. This didn’t fix it so I reverted to original Fisher Yates algorithm and experimented with different values for i. After changing i to plrow-1 the program ran successfully every time, the program had been running to many times, sometimes leading to a scenario where, players couldn’t be moved leading to an infinite loop.

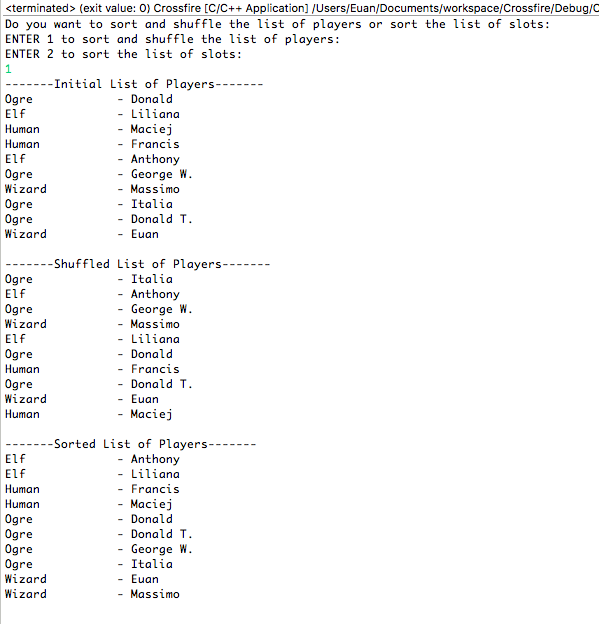




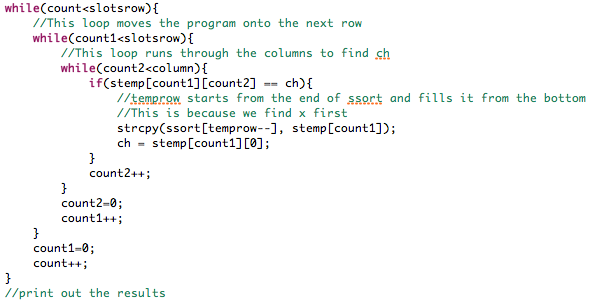
Throughout testing my shuffle algorithm I noticed a lot of random symbols when printing this, and after a lot of testing I found the problem. The struct statements I was using to transfer data in arrays from one function to another were causing issues. After learning I could just declare arrays as global arrays, I implemented this which solved the problem with any additional tinkering.

To sort the arrays the only option I saw was strcmp to compare one row of the array to the next row. This would give the us which line came first in alphabetical order then switching them if they were in the wrong order. Running this a few times would give us a solution. Using the same switching algorithm as the shuffle function, thankfully this didn’t pose many problems, as well as the new global arrays causing no issues.

This gave us the final result:



The file input for the slots text file caused no issues, copying the code from the player text file input program and changing the file directory and variable names. This only left the sorting algorithm for the slots. This function caused me the most problems, I tried many different methods to try including memcpy, memcpr, strstr, strcmp, strlen and strncmp in an effort to find a solution. The main issue to most of these programs was trying to select the correct position in the multidimensional array to see which slot followed. Rather than trying to use strlen and failing, I resulted to simply counting each element in the array until I found the correct digit which told me that was the correct row or just move on to the next row. It was much simpler to find which array came next simply using array[count][0]. I used three loops, the inner two selected the row and column, the outer loop reran the middle two loops, as due to the nature of the programme skipping rows to select the correct row, it needed to be ran multiple times to select each element with the maximum being the amount of rows-1.



This gave us the final result:

