How to Solve the n-Queens Problem

The Problem

First, let’s start with a definition of the n-Queens Problem. Given an n-by-n chessboard, how do position n queens so they can’t capture each other? More simply put, how can you position these queens so that none are on the same row, column, negative diagonal, or positive diagonal. We differentiate between negative and positive for computational ease, which will be explained later. A positive diagonal is one which points up and to the right, and a negative diagonal is one that points down and to the right. For our more concrete application, we’ll solve the 8-Queens Problem, which is how to position 8 queens on a (normal) 8x8 chessboard so they can’t capture each other. The gold standard is to find all solutions to the problem for a given n. While we won’t provide every solution, (for the 8-Queens problem there are 92 of them [CITE]) we will teach you how to find each solution.

The Solution

The most obvious solution is to simply brute force it. That is, try every single combination of 8 queens on the chessboard. Since queens are indistinguishable, and we have 81 places to put a queen on an 8x8 board, that gives us a very impressive 32,164,253,550 possible combinations. (This is known as 81 choose 8, for the mathematically inclined.) Trying to find the 92 correct combinations in the haystack of 32 billion might be difficult.

So brute force isn't incredibly helpful since it takes so long. So how can we consider all combinations without visiting them all? Well, there are combinations that contain multiple conflicts, like in Fig. 1.

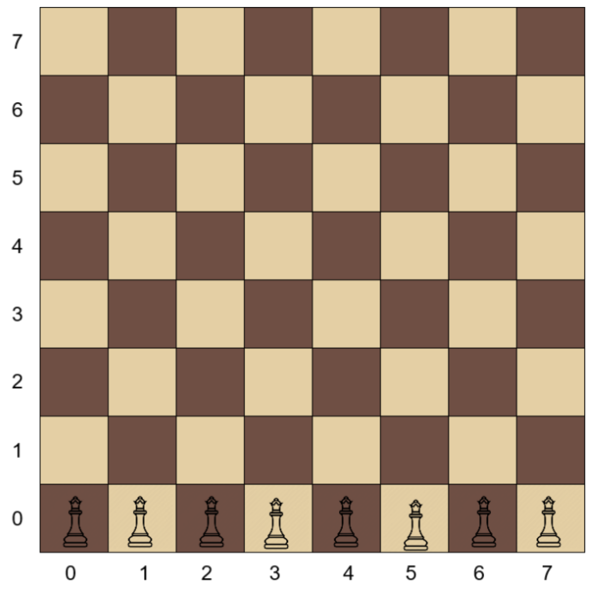


Figure 1: A possible combination of 8 queens. This is not a solution.

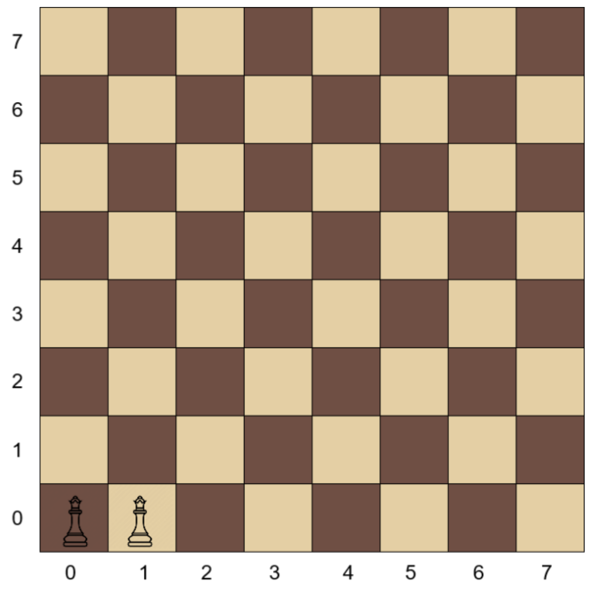


Figure 2: Any combination of 8 queens that includes these two queens in this configuration is not a solution.

Once we selected the queen in row 0, we knew we couldn't have any other queens in that row, so when the combination shows another queen in row 0, we can ignore the other 6 queens because one conflict is as bad as 8. Therefore, not only do we not consider this combination, but we also ignore any combination with any two queens in Fig. 2’s configuration. By making one comparison, we ignore 79 choose 6 (277,962,685) combinations, which is a pretty good performance gain! To make things a little simpler visually, let's assume that we correctly choose a different row for every queen we choose. So, Queen 0 (Q0) is always in row 0, Q1 in row 1, and so on. That means that for each queen/row, we only need to select a column that won't cause a conflict. So, in the following state space tree, we can see the possible choices of queens for queen 0 and queen 1 when queen 0 is in column 0.

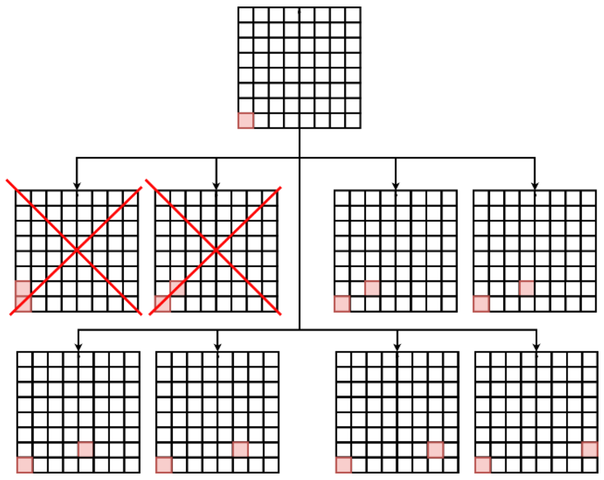


Figure 3: Graphical state space tree to choose Q1 assuming Q0 has a column of 0.

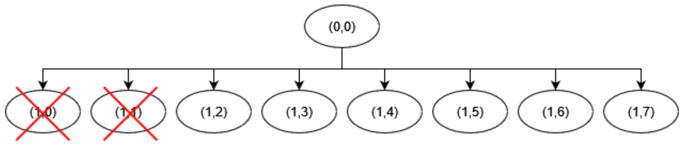


Figure 4: State space tree to choose Q1 assuming Q0 has a column of 0.

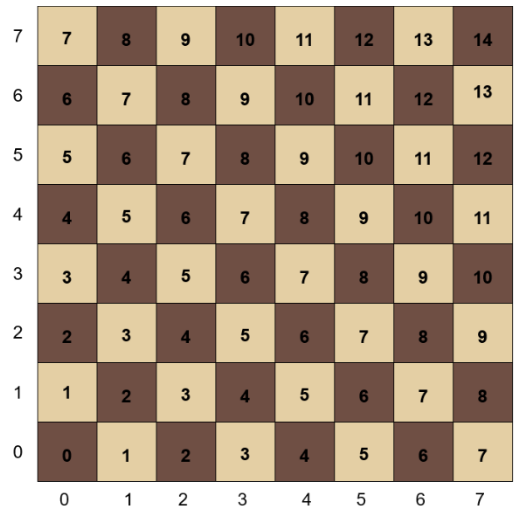


Figure 5: A chessboard with the negative diagonals labeled. (N = R + C)

In Fig. 4 is the equivalent state space tree, where each level is the queen's row, and the label on the node indicates its coordinates in (row, column), although the only change is the column. From a computational perspective, it's much easier to create the second type of state space tree. For each choice/node, you trim it (ignore subsequent choices) if any of its ancestors contain the same column. (By nature, they don't contain the same row.) This is how we check for column conflicts. But what about diagonal conflicts? For this we use a little trick. Note that both columns and rows can be represented mathematically by a combination of the row index R and the column index C. Column = 0xR + C and Row = R + 0xC. As it turns out, we can represent the positive and negative diagonals in a similar way. For negative diagonal N, note that N = R + C stays constant across the column. Note that the same is true for positive diagonal P = R – C. Thus, we can represent each queen as a tuple of 4: two independent values R (row) and C (column), and two dependent values P (positive diagonal) and N (negative diagonal.) So, Queen n Qn = (R=n, C, P, N). Just as we pruned choices with columns that conflict, we should also prune choices with positive or negative diagonals that conflict. So, this is how we construct the space state tree: at each level we attempt to make a choice and see if that attempted choice causes a conflict; in other words, we see if that node has the same column as any of its ancestors. (Note this is not just its immediate parent; we check all the way up to the first choice of Q0.) We make this same ancestral check for the diagonals. (We don't need to check the row, because our state space tree has a unique row per level by design.) This is partially illustrated in Fig. 6, which is rendered down to the Q2 level, with triangle indicating further subtrees for Q3-Q7. We simply repeat this process until we find a non-conflicting Q7. Any non-conflicting Q7, along with it's ancestors, represent a possible solution.

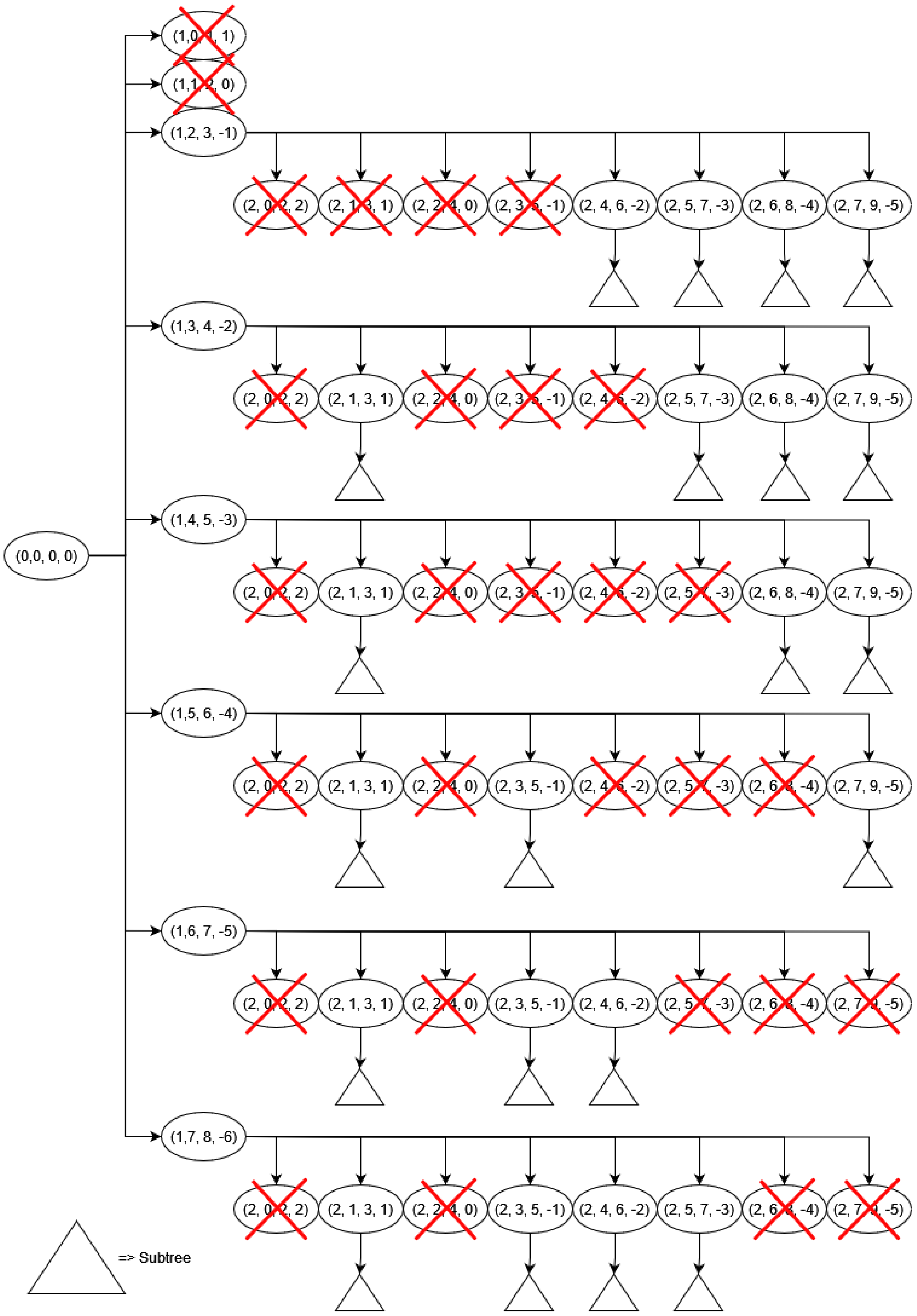


Figure 6: A state space tree to choose Q1 and Q2 assuming Q0 has a column of 0.

How to Play

This simulation will allow you to test your skills in finding solutions to the 8 Queens problem. First, click "8-Queens Game."

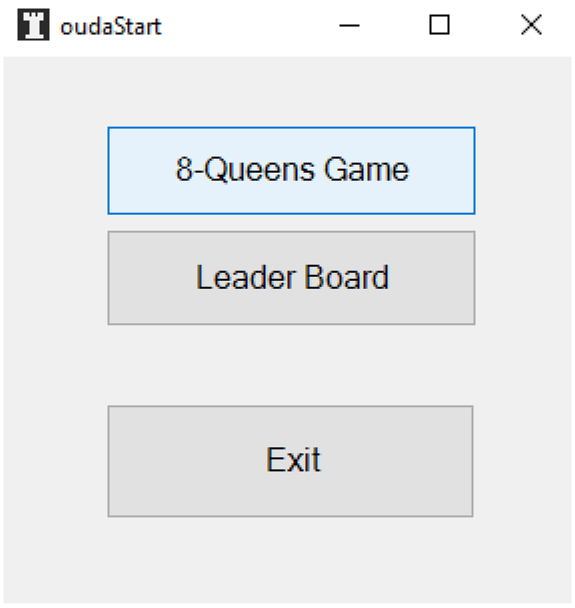


Figure 7: Start the game.

Then, you'll see a chessboard much like the one in Fig. 8. To enter a queen choice, type in a letter and a number into the text box, such as "a1," and click "Enter Queen." This will place a queen on the screen and give you points.

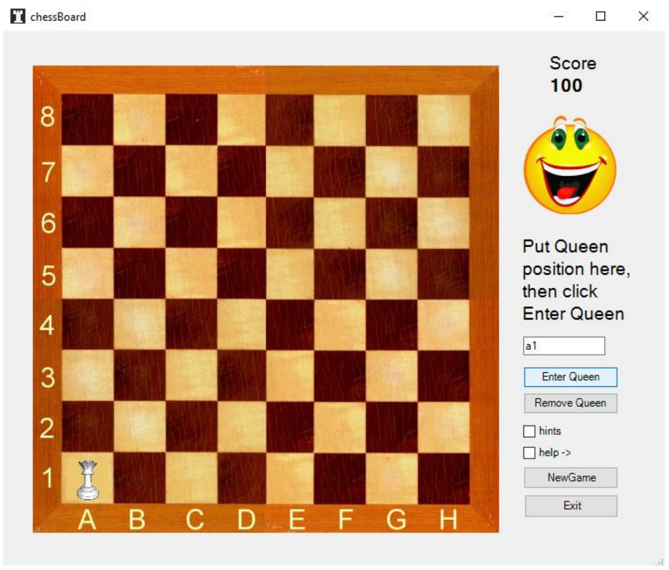
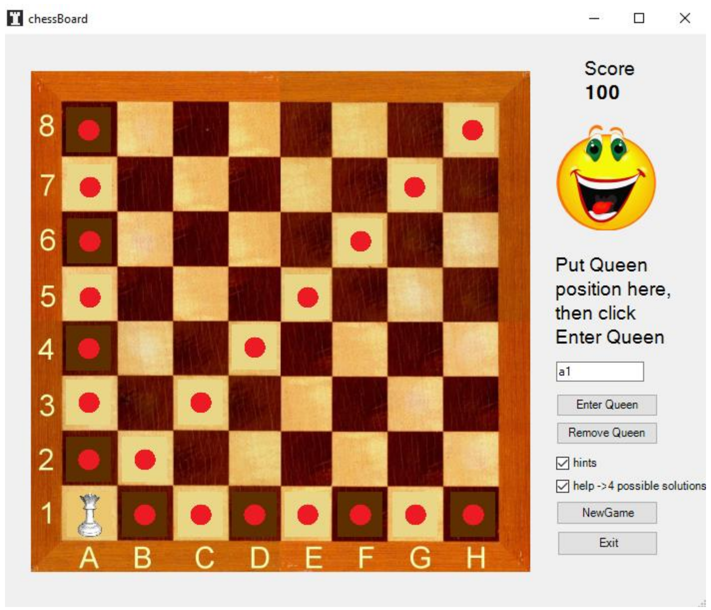


Figure 8: Place a queen.

If you’d like some extra help in seeing where NOT to place a queen, check the box labeled “hints” to see the capture paths of the queens on the board. Additionally, if you’re worried about those pruned branches we talked about earlier, check “help” and it will tell you how many possible solutions there are considering the queens you’ve placed on the board.



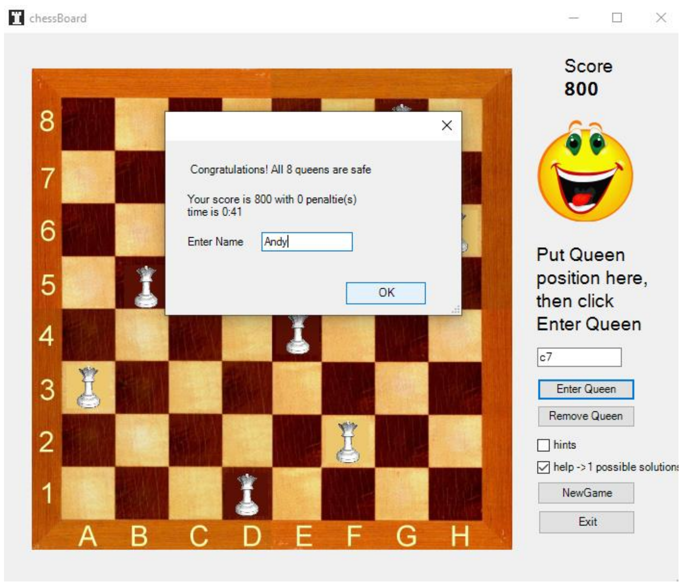
Score
100

a1

☑ hints
☑ help ->4 possible solutions

Figure 9: See capture path and how many solutions are possible with "hints" and "help."

If you place a queen incorrectly, remove it by typing its name (i.e. “a1”) into the textbox and clicking “Remove Queen.” If you manage to get all 8 queens safely on the board, congratulations! Enter your name to see it on the leaderboard. The game will then challenge you to find another solution, starting with the last queen you placed.



Score
800

Congratulations! All 8 queens are safe

Your score is 800 with 0 penaltie(s)
time is 0:41

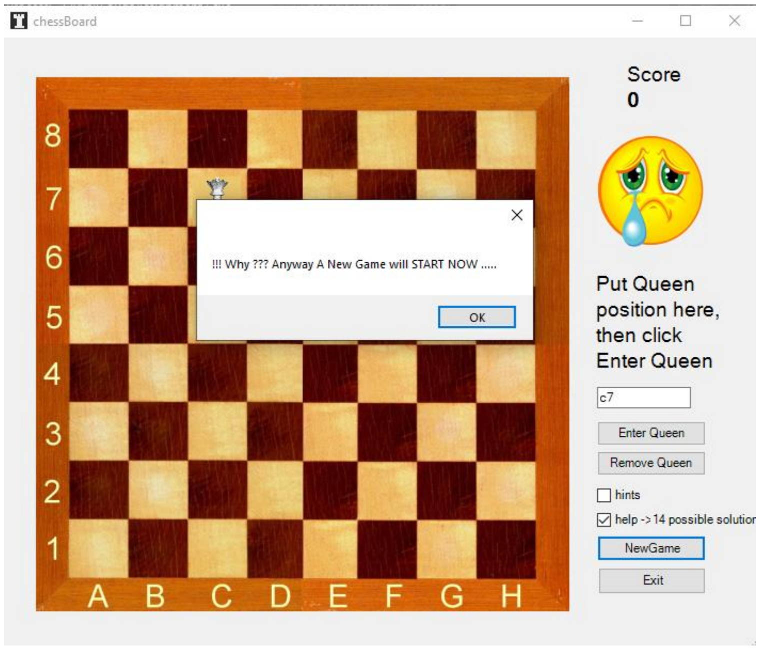
Enter Name Andy

c7

☐ hints
☑ help ->1 possible solutions

Figure 10: You've won the game!

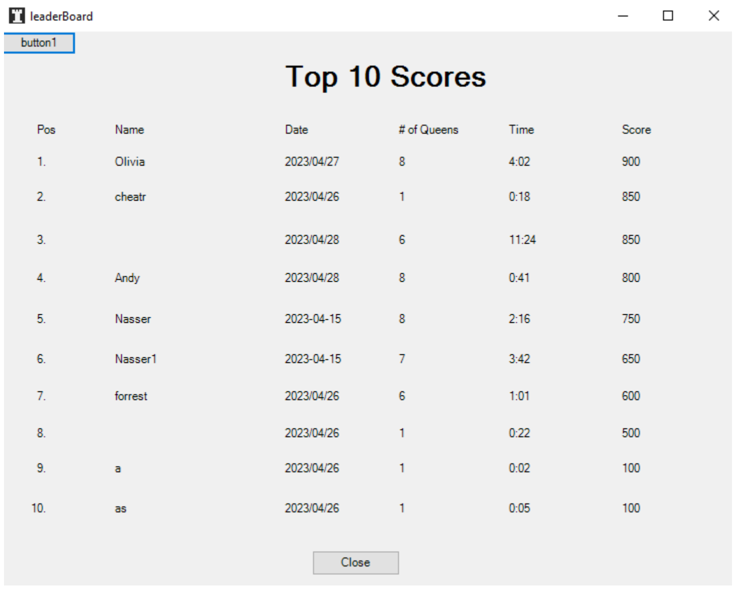
If you'd rather start from scratch, hit "NewGame."



!!! Why ??? Anyway A New Game will START NOW .....

Figure 11: Starting anew.

If you'd like to see the leaderboard, hit "Exit" and hit "Leader Board."



Top 10 Scores

Pos	Name	Date	# of Queens	Time	Score
1.	Olivia	2023/04/27	8	4:02	900
2.	cheatr	2023/04/26	1	0:18	850
3.		2023/04/28	6	11:24	850
4.	Andy	2023/04/28	8	0:41	800
5.	Nasser	2023-04-15	8	2:16	750
6.	Nasser1	2023-04-15	7	3:42	650
7.	forrest	2023/04/26	6	1:01	600
8.		2023/04/26	1	0:22	500
9.	a	2023/04/26	1	0:02	100
10.	as	2023/04/26	1	0:05	100

Figure 12: The greats.

After you're all finished up, hit "Close" and then "Exit." Thanks for playing!