https://github.com/Azubir/Domination-Game

19361113 Azubir Elsheikh

First, declared the players and then initialised by passing it into a function which assigning their relevant information such as the colour piece, asking for their name, number of pieces reserved and pieces captured.

I then declared an array of structures (board) which contains piece colour, the square type, the number of pieces in a square, top piece colour.

I have then passed the board to a function which assigned the relevant valid and invalid pieces and assigned the colour pieces to relevant squares. I used a nested loop with a couple of conditional statements which assigned invalid to certain squares and the rest were assigned valid. Certain squares were avoided and not assigned any pieces on purpose and this was done by having a condition statement which avoided the assignment of pieces to this square. Also, the colour pieces were distributed as pack of two using also conditional statements to do so.

I have then passed the board to a function which prints out the index board. It does this by using a nested loop and prints out the array element its on each time.

I have then passed the board to a function which prints out the board. I have used the function getcount which counts the number of pieces they are are and then used switch stamen which prints out he relevant information of how many pieces there are in each square. A nested loop is also used to achieve this. The condition statements were used to check the board type and for each type theirs a certain out put and then another condition which checked for the top colour on the square which is assigned to the variable I board by the array of linked list in function Playgame.

I have then declared and initialised an array of linked lists (HEAD) with assigned NULL.

I have then passed the array of linked list and the board to function which assigned the information from the board to the linked list such as the valid and invalid pieces the number of pieces in each square and the top colour.

I have then passed the HEAD, board and players to a function called Playgame which plays the game. Its for loop which loops for each player turn. I have used the modulo function to decide which player turn is. I have also given the user the option whether he wants to move a piece from the board or pieces he has reserved or to print out the first player information or the second player information. I then ask the user for the index of the piece/stack he wishes to move. I have a condition statement which checks if the piece he wished to move is a valid one if so, the game continues otherwise invalid is printed out and the other players turn. If the game continues, I ask for the index of the square he wishes to place the piece/stack. I then use a function called validity which checks whether the move is a legal move to make which depends on the size of the stack. It does this by having a number of conditional statements each statement checks for something e.g.if the row is the same, or the column is the same, or if they're both different then it uses the getcount function which computes the size of the stack and then a number of other conditional statements which contains the size of the stack and then another conditional statements which contains if he moved one step or two etc. after that theirs an array which contains all the valid moves given all the previous conditions were met and then theirs a loop which checks if theirs any one of these moves meets the criteria the player moved and if so flag is assigned 1 and loop is broken and flag is returned and assigned to g. If none of the conditions are met, then flag will remain zero and flag is returned which is assigned to g. A condition statement which checks for the value of g. If g is zero invalid is printed out and the other player move. If g is 1 the game continues. Here then I send the element of the array of linked list of both the stack that is moved and stack it is moved to a function called merge stack. This function combines the two stacks. The function it creates a linked list called list  and assigns NULL and then assigns the linked list of the piece that's being moved. I then create another linked list called curr which I assigned list to. I have while loop which traverse until the last element of the linked list. I then assign the next node of curr to the first node of the linked list which is being merged to. I then return the new merged linked list. I assign the top colour of the newly merged linked list to the relevant element of the board color. I then have a condition statement which checks whether the newly merged linked list is greater than five. I f it is greater than five I have loop which does not terminate until the linked list is back to size 5. I declare Enum color called lastcolor. I declare three linked list called curr, pre, node. I assigned the merged linked list to curr node. I then use a loop which traverse till the end of curr and the node before curr is assigned to pre. The colour at the curr is assigned to lastcolor. I then assigned NULL to pre and assigned node back to HEAD the merged linked list and this removes the last piece of the stack. I have a conditions statements to check the colour and if the colour is the same as the player the reserves are incremented otherwise the captured is incremented and this saves the last piece of the stack somewhere. I then declare a linked list and assign NULL to it, which I then assign to linked list of the square of the piece that has been moved which in this way it clears the piece from the board. I then printout the new updated board and the players updated information. If the stack is lower than size 5 then . I then printout the new updated board and the players updated information. If the player chose to move from the pieces he has reserved I then check if he has none which if its true I print out that you have none pieces reserved and then It's the next player move. If there is everything I basically send above is going to execute in the same way except for I don't ask for the index of the board for which the piece is moved from and I also decrement the players pieces reserved. The winner functionality is there but I commented it out as its not working for the meanwhile as it crashes my program rather.