Introduction

For this practical we were to implement a pointer based card trick. The trick was to work by asking the user to pick a card from a displayed list and then the program would ask the user several times which column their card was displayed in. After asking receiving the column the program would then shuffle the cards so that the column the user selected would be in the middle of the three columns, after this process was repeated three times the program was to then display the correct card to the user.

Design and implementation

The main design choice which had to be made was which approach should be taken to implement the card trick. It was decided to implement the solution as a pointer based system which would not use global variables.

One of the first design choices which had to be made was how each of the cards would be created and stored. It was decided that a struct would be used to create a new type called card which would hold a char value and a char suit value. A deck array is then populated with one of each card. For the program to simulate how the trick would work in real life the 21 cards which are used for the trick need to be shuffled. This was a tricky aspect of the program as a random card selector had to be created in order for a random selection of cards to be collected. Selecting the random cards worked by getting a random number between 0 and 51 and then selecting that card from the full deck of cards.

The main issue with this approach of selecting the cards was that there would be duplicate cards in the set of 21 which were chosen for the trick. In order to fix this issue a check had to be implemented which would loop through all of the cards that have already been selected and if the card had already been chosen then another random card would be picked until the card picked had not been chosen.

Once a card had been chosen from the full deck it needed to be stored so it could be used for the trick. It was decided to store all of the 21 values in a linked list. This storage solution was chosen as it would make it easier in the later stages of the trick when it came to splitting the 21 cards into columns and when having to re-deal the cards after the user had said which column their card was in. The linked list would store 21 pointers with each pointer representing one of the cards being used in the trick.

The linked list was created by creating a node struct which would hold a card value and which holds a pointer to the next item in the list. When it came to splitting the set of cards into their different columns the decision was made to instead of moving the values into three separate lists, a single list would be maintained with a pointer being used for the heads of each of the columns. This use of pointers helped when it came to rearranging the lists after the user had picked which column their card was in. As the next aspect of the node at the end of the list could be changed so that it pointed to the pointer which is used for the top of another column. This functionality was then used a number of times during the looping of the trick.

Testing

The following shows a run through of the program, selecting a card at the beginning and testing if the program displays the correct card at the end of the run through.

Stage	Program Output	Comment
Initial Display	Read my mind ---- Column 1 ---- Column 2 ---- Column 3 ---- QS KD 5H 2C 2S 6D JH 4C 3C QD 6S 7D AS XS XH QC 9D AC 5C 3S 8H	This is the initial displaying of the cards. There should be no duplicate cards in this set. For this test case, we will use the 5 of clubs (5C).
Input first column	---- Which column is your card in? ---- 1 ---- Column 1 ---- Column 2 ---- Column 3 ---- KD 2S 4C 6S XS 9D 3S QS 2C JH QD AS QC 5C 5H 6D 3C 7D XH AC 8H	Column 1 was selected and the new set of cards is then shown with the cards in their new columns. We can see 5C is now in column 2.
Input second column	---- Which column is your card in? ---- 2 ---- Column 1 ---- Column 2 ---- Column 3 ---- KD 6S 3S JH QC 6D XH 2S XS QS QD 5C 3C AC 4C 9D 2C AS 5H 7D 8H	Column 2 was selected and the new set is shown. Column 3 now holds 5C.
Input third column	---- Which column is your card in? ---- 3 ---- Column 1 ---- Column 2 ---- Column 3 ---- KD JH XH QS 3C 9D 5H 3S 6D XS 5C 4C AS 8H 6S QC 2S QD AC 2C 7D	Column 3 was selected and it can now be seen that 5C has been moved to the middle of column 2.
Final display with card shown	---- Your Card is: 5C	Finally, the card is displayed to the user. The program successfully predicted our card.

This testing was run a number of times to check that the program manages to display the correct card to the user.

Another aspect of the program which needed testing was the user input of which column contained their card. A check needed to implement so that only 1,2 or 3 could be accepted by the program. This check is shown below.

Input	Expected Output	Actual Output
Hello	Invalid Input! Please enter 1, 2 or 3	---- Which column is your card in? ---- ---- hello ---- Invalid Input! Please enter 1, 2 or 3 ----
50	Invalid Column! Please enter 1, 2 or 3	---- Which column is your card in? ---- ---- 50 ---- Invalid Column! Please enter 1, 2 or 3 ----
-10	Invalid Column! Please enter 1, 2 or 3	---- Which column is your card in? ---- ---- -10 ---- Invalid Column! Please enter 1, 2 or 3 ----

Difficulties

The main difficulties for this practical was working with C and understanding how to work with structs and pointers. At the beginning of the practical I did not fully understand how pointers worked however, over the course of implementing the practical I began to learn how they worked and how the best way was to use them when implementing solutions.

I also struggled at points when implementing the collecting of each of the columns and then re-dealing them in the correct order. This took time to understand and the solution which I managed to implement could be better optimised.

Conclusion

Overall, I managed to design and implement the ‘read my mind’ card trick, as the program displays a random set of 21 cards to the user with no duplicate cards being present in this set. The user is then asked which column their card is present in, with the cards then being shuffled and displayed in the correct way for the trick to work. After this process is repeated a number of times the correct card is then displayed to the user. The trick was implemented by using pointers and linked lists throughout the program while not using global variables in order for the program to work correctly.

If I were to spend more time on the practical I would spend the main focus on trying to optimise the program more so there is less checks taking place in loops and I would look at trying to improve the shuffling and checking if a card had already been drawn as this is currently very inefficient. The amount of code which is currently present in the main method would also be cut down substantially as currently there is more implemented in the main method then would normally be considered acceptable.

Extensions

The first extension which was implemented was allowing the user to play another go of the trick after they have finished one. After the user has completed the trick and their card has been displayed to them they are now promoted if they would like to do the trick again. This feature was also implemented with input validation to make sure only 'Y' or 'N' could be entered. The testing for this extension is show below.

Input	Expected Output	Actual Output
Y	New game begins	---- Would you like to play again? (Y/N) ---- ---- Y ---- Column 1 ---- ---- Column 2 ---- ---- Column 3 ---- 4 of Hearts 3 of Hearts J of Spades 2 of Clubs 2 of Spades J of Hearts 5 of Hearts 6 of Diamonds 9 of Spades Q of Spades 2 of Diamonds 3 of Clubs J of Diamonds A of Diamonds 6 of Hearts K of Spades 8 of Clubs 7 of Clubs 3 of Spades 5 of Clubs A of Clubs ---- Which column is your card in? ---- ----
N	Game exits	---- Would you like to play again? (Y/N) ---- ---- N ---- Thank you for playing! ----
Hello123	Invalid Input! Please enter Y to play again or N to quit	---- Would you like to play again? (Y/N) ---- ---- hello123 ---- Invalid Input! Please enter Y to play again or N to quit ---- ----

The second extension which was to implement a better UI for the cards suits. Before each of the suits was represented by a single character such as 'H' for hearts. Although this implementation worked it was not the most appealing way of implementing suits. To fix this issue the storage of the suits was changed from chars to suits which allowed for the suits to be referred to by their name instead of by a single letter.

After implementing the suits, a change was then made to the storage of the values of the cards. Before each of the values were stored again as chars, however this meant that any card with a value of ten had to be represented by a 'X'. By changing the values to strings it meant that a ten card could instead be represented by '10' which is how it would be in a real-life pack of cards.

```
---- Column 1 ----    ---- Column 2 ----    ---- Column 3 ----
     8  of Hearts          J  of Clubs           K  of Diamonds
     K  of Clubs           A  of Hearts          Q  of Clubs
     6  of Diamonds        4  of Spades          5  of Hearts
     Q  of Hearts          5  of Clubs           7  of Spades
     7  of Hearts          8  of Clubs           2  of Clubs
     J  of Spades          9  of Hearts          9  of Diamonds
     4  of Clubs           7  of Clubs           A  of Spades
```