```
                                  ---CARDS---

              everything to do with working with cards and decks

+------------------------------------------------------------------------------+
|                                    FILES                                     |
+------------------------------------------------------------------------------+


cards.h

      type definitions and function prototypes for the exposed interface for
            working with cards and decks


cards.c

      implementation of card and deck functions


rendering.h

      function prototypes for the exposed interface for working with rendering
            decks as ascii art


rendering.c

      implementation of ascii art rendering functions for decks








+------------------------------------------------------------------------------+
|                                    TYPES                                     |
+------------------------------------------------------------------------------+


typedef struct card card;
struct card{
    char suit[10];
    int value;
    card *next;
    card *prev;
};
```

```
      a card in a deck
      suit is one of "hearts", "spades", "clubs", or "diamonds"
      value is in the interval [1, 13]
      values 1-10 represent the integers 1-10
      values 11, 12, and 13 represent jack, queen, and king, respectively
      each card is a node in a linked list
```

```
+------------------------------------------------------------------------------+
|                                  FUNCTIONS                                   |
+------------------------------------------------------------------------------+
```

cards.h

```
      card *newcard(char suit[7] , int value);

      card *newdeck(void);

      card *removeCard(card *head , card *cardToBeRemoved);

      card *getLastCard(card *head);

      card *getCard(card *head,int index);

      card *shuffleDeck(card *head);

      int deckSize(card *head);

      void printCard(card *c);

      void printDeck(card *c);

      void freeDeck(card *head);

      void appendCard(card *head, card*c);

      void predefinedCards(void);
```

rendering.h

```
      void printDeckFancy(card *c);
```

```
+-----------------------------------------------------------------------------+
|                             FUNCTIONS -- IN DETAIL                          |
+-----------------------------------------------------------------------------+

                                  ---cards.h---

card *newcard(char suit[7] , int value);

      allocate memory for a new card instance

      Parameters:
            suit -- the suit of the card(hearts, spades, clubs, or diamonds)
            value -- the value of the card(1, 2, 3, etc.)

      Returns:
            a pointer to a newly allocated card

      Pre-conditions:
            suit is "hearts", "spades", "clubs", or "diamonds"
            value is in the interval [1, 13]

      Post-conditions:
            memory is dynamically allocated for a new card
            this memory will need to be freed later on


card *newdeck(void);

      create a new deck

      Parameters:
            void

      Returns:
            a pointer to the first card in the newly allocated deck

      Pre-conditions:
            none

      Post-conditions:
            memory is dynamically allocated for a deck of 52 cards
            this memory will need to be freed later on
```

```
card *removeCard(card *head , card *cardToBeRemoved);

      unlink a card from the deck

      Parameters:
            head -- a pointer to the first card in the deck
            cardToBeRemoved -- a pointer to the card to remove from the deck

      Returns:
            a pointer to the new first card in the deck
            this is necessary because it is possible that the first card in the
                  deck could be removed(if head and cardToBeRemoved point to the
                  same card)
            this function should be called as follows:
                  deckHead = removeCard(deckHead, cardToRemove);
                  being sure to reassign the head in case it changes
            this is done instead of using a double pointer

      Pre-conditions:
            head is not NULL
            cardToBeRemoved is not NULL

      Post-conditions:
            cardToBeRemoved is removed from the deck pointed to by head
            a pointer to the first card in the deck is returned


card *getLastCard(card *head);

      get a pointer to the last card in the deck

      Parameters:
            head -- a pointer to the first card in the deck

      Returns:
            a pointer to the last card in the deck

      Pre-conditions:
            head is not NULL

      Post-conditions:
            a pointer to the last card in the deck is returned


card *predifinedCards(void);

      load a predefined deck from a file

      Parameters:
            void
      Returns:
            a pointer to the first card in the deck

      Pre-conditions:
            the file "cards.txt" exists

      Post-conditions:
            a deck is loaded from the file


card *getCard(card *head,int index);

      get a pointer to a card in the deck at the specified index

      Parameters:
            head -- a pointer to the first card in the deck
            index -- the index of the desired card

      Returns:
            a pointer to the card at the specified index

      Pre-conditions:
            index does not exceed one less than the size of the deck

      Post-conditions:
            a pointer to the desired card is returned


card *shuffleDeck(card *head);

      shuffle the deck

      Parameters:
            head -- a pointer to the first card in the deck

      Returns:
            a pointer to the first card in the shuffled deck
            this could be different from the first card before the deck was
                  shuffled, as that card might not be first anymore

      Pre-conditions:
            head is not NULL
            there are 52 cards in the deck pointed to by head

      Post-conditions:
            the deck pointed to by head is shuffled, and a pointer to the new head
                  is returned


int deckSize(card *head);
```

```
      count how many cards are in the deck

      Parameters:
            head -- a pointer to the first card in the deck

      Returns:
            the number of cards in the deck pointed to by head

      Pre-conditions:
            none

      Post-conditions:
            the number of cards in the deck pointed to by head is returned


void printCard(card *c);

      print a card to the console

      Parameters:
            c -- a pointer to the card to be printed

      Returns:
            void

      Pre-conditions:
            c is not NULL

      Post-conditions:
            a card is printed to the console


void printDeck(card *c);

      print a deck to the console

      Parameters:
            c -- a pointer to the first card in the deck pointed to by c

      Returns:
            void

      Pre-conditions:
            c is not NULL
            there are 52 cards in the deck pointed to by c

      Post-conditions:
            the deck is printed to the console


void freeDeck(card *head);
```

```
      free the memory used by each card in the deck

      Parameters:
            head -- a pointer to the first card in the deck

      Returns:
            void

      Pre-conditions:
            all of the non-NULL pointers in the linked list of cards point to
                  memory that has been allocated but has not yet been freed

      Post-conditions:
            the memory used by each card in the deck is freed


void appendCard(card *head, card*c);

      add a card to the end of the deck

      Parameters:
            head -- a pointer to the first card in the deck
            c -- a pointer to the card to be added to the end of the deck

      Returns:
            void

      Pre-conditions:
            head is not NULL
            c is not NULL
            a card cannot be appended to an empty deck using this function

      Post-conditions:
            c is added to the end of the deck pointed to by head


                                    ---rendering.h---

void printDeckFancy(card *head);

      print a deck to the console using ascii art

      Parameters:
            head -- a pointer to the first card in the deck

      Returns:
            void

      Pre-conditions:
            none

      Post-conditions:
```

the deck pointed to by head is printed to the console using ascii art