```
 1: /*
 2:  * shuffle.cpp
 3:  *
 4:  *  Created on: Jan 30, 2017
 5:  *      Author: Brian Clinkenbeard
 6:  */
 7:
 8: #include "header.h"
 9: #include "card.h"
10:
11: /* 1b. perform a perfect shuffle */
12: void shuffleDeck(card shuffle[52])
13: {
14:         /* preserve original deck (cannot copy array) */
15:         card unshuffled[52];
16:         for (int i = 0; i < 52; ++i) {
17:                 unshuffled[i] = shuffle[i];
18:         }
19:
20:         /* perfect shuffle */
21:         int firstHalf = 0;
22:         int secondHalf = 26;
23:         int count = 0;
24:         while (count < 51) {
25:                 shuffle[count] = unshuffled[firstHalf];
26:                 firstHalf++;
27:                 count++;
28:                 shuffle[count] = unshuffled[secondHalf];
29:                 secondHalf++;
30:                 count++;
31:         }
32: }
33:
34:
```

```
    1: /*
    2:  * main.cpp
    3:  *
    4:  *  Created on: Jan 30, 2017
    5:  *      Author: Brian Clinkenbeard
    6:  */
    7:
    8: #include "header.h"
    9: #include "card.h"
   10:
   11: int main()
   12: {
   13:         /* 1b. initialize deck */
   14:         card deck[52];
   15:
   16:         /* iterate through the enums */
   17:         int count = 0;
   18:         for (int i = 0; i < 4; ++i) {
   19:                 suit initSuit = static_cast<suit>(i);
   20:                 for (int j = 0; j < 13; ++j) {
   21:                         face initFace = static_cast<face>(j);
   22:                         deck[count] = card(initFace, initSuit);
   23:                         count++;
   24:                 }
   25:         }
   26:
   27:         /* perfect shuffle */
   28:         card shuffled[52] = deck;
   29:         shuffleDeck(shuffled);
   30:
   31:
   32:         /* shuffle until deck is returned to original */
   33:         card final[52] = shuffled;
   34:         count = 1; /* one shuffle has already occurred */
   35:         while (!equalDecks(deck, final)) {
   36:                 shuffleDeck(final);
   37:                 count++;
   38:         }
   39:
   40:         /* 1c. print decks */
   41:         cout << "Initial deck:" << endl;
   42:         printDeck(deck);
   43:         cout << "Shuffled deck:" << endl;
   44:         printDeck(shuffled);
   45:         cout << "Final deck:" << endl;
   46:         printDeck(final);
   47:
   48:         /* 1d. print amount of shuffles */
   49:         cout << "It took " << count << " perfect shuffles for the deck to return t
o its original order." << endl;
   50:
   51:         return 0;
   52: }
```

```
 1: /*
 2:  * card.cpp
 3:  *
 4:  *  Created on: Jan 30, 2017
 5:  *      Author: Brian Clinkenbeard
 6:  */
 7:
 8: #include "header.h"
 9: #include "card.h"
10:
11: /* for array instantiation */
12: card::card() {}
13:
14: /* constructor */
15: card::card(face newFace, suit newSuit)
16: {
17:         myFace = newFace;
18:         mySuit = newSuit;
19: }
20:
21: /* "break;" not necessary on returns */
22: string card::getFace()
23: {
24:         switch (myFace) {
25:         case ACE:
26:                 return "Ace";
27:         case TWO:
28:                 return "2";
29:         case THREE:
30:                 return "3";
31:         case FOUR:
32:                 return "4";
33:         case FIVE:
34:                 return "5";
35:         case SIX:
36:                 return "6";
37:         case SEVEN:
38:                 return "7";
39:         case EIGHT:
40:                 return "8";
41:         case NINE:
42:                 return "9";
43:         case TEN:
44:                 return "10";
45:         case JACK:
46:                 return "Jack";
47:         case QUEEN:
48:                 return "Queen";
49:         case KING:
50:                 return "King";
51:         }
52: }
53:
54: string card::getSuit()
55: {
56:         switch (mySuit) {
57:         case CLUBS:
58:                 return "Clubs";
59:         case DIAMONDS:
60:                 return "Diamonds";
61:         case HEARTS:
62:                 return "Hearts";
63:         case SPADES:
64:                 return "Spades";
65:         }
66: }
67:
68: /* print the face and suit of the card */
69: ostream& operator<<(ostream &os, card &outcard)
70: {
```

```
   71:         os << outcard.getFace() << " of " << outcard.getSuit();
   72:         return os;
   73: }
   74:
   75: /* overload relational operators for comparison */
   76: bool operator==(const card &firstCard, const card &secondCard)
   77: {
   78:         return (firstCard.myFace == secondCard.myFace && firstCard.mySuit == secon
dCard.mySuit);
   79: }
   80:
   81: bool operator!=(const card &firstCard, const card &secondCard) {
   82:         return !(firstCard == secondCard);
   83: }
```

```
 1: /*
 2:  * print.cpp
 3:  *
 4:  *  Created on: Jan 30, 2017
 5:  *      Author: Brian Clinkenbeard
 6:  */
 7:
 8: #include "header.h"
 9: #include "card.h"
10:
11: /* 1b. print the deck of cards */
12: void printDeck(card printDeck[52])
13: {
14:         for (int i = 0; i < 52; ++i) {
15:                 cout << i + 1 << ". " << printDeck[i] << endl;
16:         }
17: }
```

```
 1: /*
 2:  * compare.cpp
 3:  *
 4:  *  Created on: Jan 30, 2017
 5:  *      Author: Brian Clinkenbeard
 6:  */
 7:
 8: #include "header.h"
 9: #include "card.h"
10:
11: /* 1b. compare two decks of cards */
12: bool equalDecks(const card firstDeck[52], const card secondDeck[52])
13: {
14:         for (int i = 0; i < 52; ++i) {
15:                 if (firstDeck[i] != secondDeck[i])
16:                         return false;
17:         }
18:         /* fall-through: all cards are equal */
19:         return true;
20: }
21:
22:
```