1. （洗牌和發牌）修改圖 7.24 的程式，使發牌函式能夠一次發出 5 張牌。然後寫出以下的函式：

   a) 判斷這 5 張牌裡是否有對子。

   b) 判斷這 5 張牌裡是否有雙對子。

   c) 判斷這 5 張牌裡是否有三條（如 3 張傑克）。

   d) 判斷這 5 張牌裡是否有鐵支（如 4 張 A）。

   e) 判斷這 5 張牌裡是否有同花（即 5 張同樣花色的牌）。

   f) 判斷這 5 張牌裡是否有順子（即 5 張連續的牌）。

牌型 (別名)	英文名	說明	範例				
同花順	Straight Flush	五張同一花色且順連的牌。	10♠	J♠	Q♠	K♠	A♠
四條	Four of a Kind	有四張同一點數的牌。	4♠	4♥	4♣	4♦	9♥
葫蘆	Full house	三張同一點數的牌，加一對其他點數的牌。	8♠	8♣	8♦	K♠	K♥
同花	Flush	五張同一花色的牌。	3♠	4♠	8♠	J♠	K♠
順子	Straight	五張順連的牌。	A♣	2♣	3♥	4♦	5♠

三條	Three of a kind	有三張同一點數的牌。	7♠	7♥	7♦	2♣	K♦
兩對	Two Pairs	有兩張相同點數的牌，加另外兩張相同點數的牌。	8♠	8♦	A♥	A♣	Q♠
一對	One Pair	有兩張相同點數的牌。	9♠	9♥	4♣	J♠	A♥
散牌	High card	不能排成以上組合的牌，以點數決定大小。	3♠	6♥	9♦	K♠	A♣

如果有兩個玩家，請判定那一家獲勝。

```
// Fig. 7.24: fig07_24.c
// Card shuffling and dealing.
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define SUITS 4
#define FACES 13
#define CARDS 52

// prototypes
void shuffle( unsigned int wDeck[][ FACES ] ); // shuffling modifies wDeck
void deal( unsigned int wDeck[][ FACES ], const char *wFace[],
   const char *wSuit[] ); // dealing doesn't modify the arrays

int main( void )
{
```

```c
   // initialize suit array
   const char *suit[ SUITS ] =
      { "Hearts", "Diamonds", "Clubs", "Spades" };

   // initialize face array
   const char *face[ FACES ] =
      { "Ace", "Deuce", "Three", "Four",
        "Five", "Six", "Seven", "Eight",
        "Nine", "Ten", "Jack", "Queen", "King" };

   // initialize deck array
   unsigned int deck[ SUITS ][ FACES ] = { 0 };

   srand( time( NULL ) ); // seed random-number generator

   shuffle( deck ); // shuffle the deck
   deal( deck, face, suit ); // deal the deck
} // end main

// shuffle cards in deck
void shuffle( unsigned int wDeck[][ FACES ] )
{
   size_t row; // row number
   size_t column; // column number
   size_t card; // counter


      // place card number in chosen slot of deck
      wDeck[ row ][ column ] = card;
   } // end for
} // end function shuffle

// deal cards in deck
void deal( unsigned int wDeck[][ FACES ], const char *wFace[],
   const char *wSuit[] )
{
   size_t card; // card counter
   size_t row; // row counter
   size_t column; // column counter

   // deal each of the cards
   for ( card = 1; card <= CARDS; ++card ) {
      // loop through rows of wDeck
      for ( row = 0; row < SUITS; ++row ) {
         // loop through columns of wDeck for current row
         for ( column = 0; column < FACES; ++column ) {
            // if slot contains current card, display card
            if ( wDeck[ row ][ column ] == card ) {
               printf( "%5s of %-8s%c", wFace[ column ], wSuit[ row ],
                  card % 2 == 0 ? '\n' : '\t' ); // 2-column format
            } // end if
         } // end for
      } // end for
   } // end for
} // end function deal
```

2. （使用函式指標的計算器）使用圖 7.28 中學到的技巧，建立一個文字型態的選單式程式，讓使用者選擇加減乘除兩個數字。接著，程式應該讓使用者輸入兩個 double 值，執行適當的計算並顯示結果。利用一組函式指標陣列，其中每個指標表示一個函式，該函式回傳 void，並接收兩個 double 參數。相對應的函式應該分別顯示指定要執行哪一種計算的訊息、參數值和計算結果。

3.（利用函式指標計算圓形周長、圓形面積或球體體積）利用你在圖 7.28 學到的技巧，建立一個文字型態的選單式程式，讓使用者選擇要計算圓形周長、圓形面積或球體體積。接著，程式應該讓使用者輸入半徑，執行適當的計算並顯示結果。請使用一組函式指標，每個指標表示一個回傳 void 並接收 double 參數的函式。每個相對應的函式應該顯示指定要執行哪一種計算的訊息、半徑的值、參數的值，以及計算結果。

圓周長 = 直徑$\times 3.14$

圓面積 = 半徑$\times$半徑$\times 3.14$

球體體積$=4/3\times 3.14\times$半徑^3

1. ***(Card Shuffling and Dealing)*** Modify the program in Fig. 7.24 so that the card-dealing function deals a five-card poker hand. Then write the following additional functions:
   a) Determine whether the hand contains a pair.
   b) Determine whether the hand contains two pairs.
   c) Determine whether the hand contains three of a kind (e.g., three jacks).
   d) Determine whether the hand contains four of a kind (e.g., four aces).
   e) Determine whether the hand contains a flush (i.e., all five cards of the same suit).
   f) Determine whether the hand contains a straight (i.e., five cards of consecutive face values).

牌型 (別名)	英文名	說明	範例				
同花順	Straight Flush	五張同一花色且順連的牌。	10♠	J♠	Q♠	K♠	A♠
四條	Four of a Kind	有四張同一點數的牌。	4♠	4♥	4♣	4♦	9♥
葫蘆	Full house	三張同一點數的牌，加一對其他點數的牌。	8♠	8♣	8♦	K♠	K♥
同花順	Flush	五張同一花色的牌。	3♠	4♠	8♠	J♠	K♠
順子	Straight	五張順連的牌。	A♣	2♣	3♥	4♦	5♠
三條	Three of a kind	有三張同一點數的牌。	7♠	7♥	7♦	2♣	K♦
兩對	Two Pairs	有兩張相同點數的牌，加另外兩張相同點數的牌。	8♠	8♦	A♥	A♣	Q♠

一對	One Pair	有兩張相同點數的牌。	9♠	9♥	4♣	J♠	A♥
散牌	High card	不能排成以上組合的牌，以點數決定大小。	3♠	6♥	9♦	K♠	A♣

If there are two player, the program can determine which is the winner.

```c
// Fig. 7.24: fig07_24.c
// Card shuffling and dealing.
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define SUITS 4
#define FACES 13
#define CARDS 52

// prototypes
void shuffle( unsigned int wDeck[][ FACES ] ); // shuffling modifies wDeck
void deal( unsigned int wDeck[][ FACES ], const char *wFace[],
   const char *wSuit[] ); // dealing doesn't modify the arrays

int main( void )
{
   // initialize suit array
   const char *suit[ SUITS ] =
      { "Hearts", "Diamonds", "Clubs", "Spades" };

   // initialize face array
   const char *face[ FACES ] =
      { "Ace", "Deuce", "Three", "Four",
        "Five", "Six", "Seven", "Eight",
        "Nine", "Ten", "Jack", "Queen", "King" };

   // initialize deck array
   unsigned int deck[ SUITS ][ FACES ] = { 0 };

   srand( time( NULL ) ); // seed random-number generator

   shuffle( deck ); // shuffle the deck
   deal( deck, face, suit ); // deal the deck
} // end main

// shuffle cards in deck
void shuffle( unsigned int wDeck[][ FACES ] )
{
   size_t row; // row number
   size_t column; // column number
   size_t card; // counter


      // place card number in chosen slot of deck
      wDeck[ row ][ column ] = card;
   } // end for
} // end function shuffle

// deal cards in deck
void deal( unsigned int wDeck[][ FACES ], const char *wFace[],
   const char *wSuit[] )
{
   size_t card; // card counter
   size_t row; // row counter
   size_t column; // column counter

   // deal each of the cards
   for ( card = 1; card <= CARDS; ++card ) {
      // loop through rows of wDeck
      for ( row = 0; row < SUITS; ++row ) {
         // loop through columns of wDeck for current row
         for ( column = 0; column < FACES; ++column ) {
            // if slot contains current card, display card
            if ( wDeck[ row ][ column ] == card ) {
               printf( "%5s of %-8s%c", wFace[ column ], wSuit[ row ],
                  card % 2 == 0 ? '\n' : '\t' ); // 2-column format
            } // end if
         } // end for
      } // end for
   } // end for
} // end function deal
```

2. ***(Calculator Using Function Pointers)*** Using the techniques you learned in Fig. 7.28, create a text-based, menu-driven program that allows the user to choose whether to add, subtract, multiply or divide two numbers. The program should then input two double values from the user, perform the appropriate calculation and display the result. Use an array of function pointers in which each pointer represents a function that returns void and receives two double parameters. The corresponding functions should each display messages indicating which calculation was performed, the values of the parameters and the result of the calculation.

3. ***(Calculating Circle Circumference, Circle Area or Sphere Volume Using Function Pointers)*** Using the techniques you learned in Fig. 7.28, create a text-based, menu-driven program that allows the user to choose whether to calculate the circumference of a circle, the area of a circle or the volume of a sphere. The program should then input a radius from the user, perform the appropriate calculation and display the result. Use an array of function pointers in which each pointer represents a function that returns void and receives a double parameter. The corresponding functions should each display messages indicating which calculation was performed, the value of the radius and the result of the calculation.

Circle Circumference = $2 \times \text{radius} \times 3.14$

Circle Area = $\text{radius} \times \text{radius} \times 3.14$

Sphere Volume= $4/3 \times 3.14 \times \text{radius}^3$