Arrays

Practice Problems:

Your instructor may have made the following project available to you online:

CPractice-Arrays

If not, create your own project, copy the following into it, and solve the problems in that project.

```
/*******************************************************************************
 Practice problems for:
   -- Arrays

 The first problem (TODO's 2 and 1) also provides practice for:
   -- Getting Input From the User by Using scanf.

 The second problem (TODO's 4 and 3) also provides practice for:
   -- Using Pointers to Send Information Back From a Function.

 There is also a third problem (TODO's 6 and 5).

 Problems designed by David Mutchler, July 2010.
 *******************************************************************************/

#include <stdio.h>
#include <stdlib.h>

void fillArrayWithRandomFloats(float numbers[], int arrayLength);
void fillArrayWithRandomInts(int numbers[], int arrayLength, int seed);

int main() {
    /***************************************************************************
         *** TODO 2 ***
     Write code here that tests
         fillArrayFromInput (which you write below),
     as follows:  Your code here should simply call   arrayFillFromInput.
     Examine the output that it produces to confirm that it worked correctly.
     ***************************************************************************/

    /***************************************************************************
         *** TODO 4 ***
     Write code here that tests
         twoSums (which you write below),
     as follows:  Your code here should do at least these two tests:

     Test 1:
       -- Declare an array of 6 float's.
       -- Initialize the array to contain these 6 numbers:
            3.0   4.5   -1.0    9.0    -15.0    8.0
       -- Call twoSums, sending it your array and other appropriate arguments.
       -- Print the information sent back by twoSums, namely:
          -- the sum of the positive numbers in the array
          -- the sum of the negative numbers in the array

       If your code is correct, the numbers printed should be:
          24.500000   -16.000000
 Test 2:
   -- Declare an array of 100 float's.
   -- Call   fillArrayWithRandomFloats   (provided below) with appropriate
      arguments to fill your array with 100 random float's.
   -- Call twoSums, sending it your array and other appropriate arguments.
   -- Print the information sent back by twoSums, namely:
      -- the sum of the positive numbers in the array
      -- the sum of the negative numbers in the array

   If your code is correct, the numbers printed should be:
      13.739065     -11.117590
   or possibly slightly different numbers, like
      13.739067     -11.117588
   depending on whether or not you used double precision internally
   to compute your sums (either approach is acceptable here).
 *****************************************************************************/

/******************************************************************************
     *** TODO 6 ***
 Write code here that tests
     twoInARow (which you write below),
 as follows:  Your code here should do at least these three tests:

 Test 1:
   -- Declare an array of 6 int's.
   -- Initialize the array to contain these 6 numbers:
         7  8  -18   12  12  102
   -- Call twoInARow, sending it your array and other appropriate argument(s).
   -- Print the number returned by twoInARow
      [if your code is correct, that number will be 1]

 Test 2:
   -- As in Test 1, but declare another array and use these 6 numbers:
         7  8  -18   7  8  7
      [if your code is correct, the number printed will be 0]

 Test 3
   1. Declare an array of 15 int's.

   2. a. Call   fillArrayWithRandomInts   (provided below), sending it your
         array, its length (15), and 0 as the "seed" for the
         random-number generator.
     b. Call twoInARow, sending it your array and other appropriate
        argument(s).
      c. Print the number returned by twoInARow.
         [if your code is correct, that number printed will be 0]

   3. Repeat step 2 nine more times, using the following seeds:
       when the seed is 1, your code should print 0
       when the seed is 2, your code should print 0
       when the seed is 3, your code should print 1
       when the seed is 4, your code should print 0
       when the seed is 5, your code should print 1
       when the seed is 6, your code should print 0
       when the seed is 7, your code should print 1
```

```
        when the seed is 8, your code should print 1
        when the seed is 9, your code should print 1
    ***************************************************************************/

   return EXIT_SUCCESS;
}

/****************************************************************************
 USE this function called
     fillArrayWithRandomFloats
 when you do TODO 4.
 This function fills the array that it is given with pseudo-random numbers,
 each of which is between 0 and 1.

 The function sets the random-number generator's "seed" to a fixed number,
 so that the array is filled with the same set of pseudo-random numbers each
 time this function is called, to assist testing.

 Do NOT modify this function.
****************************************************************************/
void fillArrayWithRandomFloats(float numbers[], int arrayLength) {
   int k;

   srand(10000);

   for (k = 0; k < arrayLength; ++k) {
         numbers[k] = ((float) rand() / (float) RAND_MAX) - 0.5;
         // printf("%f ", numbers[k]);
   }
   // printf("\n");
}

/****************************************************************************
 USE this function called
     fillArrayWithRandomInts
 when you do TODO 6.
 This function fills the array that it is given with pseudo-random numbers,
 each of which is between 0 and 24, inclusive.

 The function uses the given "seed" as the random-number generator's "seed".
 Calling this function with a fixed seed always fills the array with the same
 set of pseudo-random numbers, to assist testing.

 Do NOT modify this function.
****************************************************************************/
void fillArrayWithRandomInts(int numbers[], int arrayLength, int seed) {
   int k;

   srand(seed);

   for (k = 0; k < arrayLength; ++k) {
         numbers[k] = rand() % 25;
         // printf("%3i", numbers[k]);
   }
   // printf("\n");
}
```

```
/*****************************************************************************
    *** TODO 1 ***
 Write a function called
     fillArrayFromInput
 that takes NO arguments.  The function:
   a. Asks the user how many floating-point numbers she will enter.
   b. Gets that many floating-point numbers from the user.
   c. Prints the floating point numbers in the reverse order in which
      they were entered, all on one line.
 Note that you need an array to solve this problem.

 Here is a sample run:
   How many floating-point numbers will you enter? 5
   Enter a floating-point number: 4.9
   Enter a floating-point number: -56.3
   Enter a floating-point number: 18
   Enter a floating-point number: 0
   Enter a floating-point number: 3.14159
   3.141590 0.000000 18.000000 -56.300000 4.900000
*****************************************************************************/

/*****************************************************************************
    *** TODO 3 ***
 Write a function called
     twoSums
 that takes FOUR arguments:
    -- an array of float's
    -- the length of the array
    -- two pointers to float's.
 The function uses the last two pointer parameters to send back to the caller:
    -- the sum of the positive numbers in the array, and
    -- the sum of the negative numbers in the array.

 For example, if the array contains these 6 numbers:
   3.0  4.5  -1.0   9.0   -15.0   8.0
 then the first of the two pointee's should be set to 24.5
   (which is 3.0 + 4.5 + 9.0 + 8.0)
 and the second of the two pointee's should be set to -16.0
   (which is -1.0 + -15.0).
*****************************************************************************/

/*****************************************************************************
    *** TODO 5 ***
 Write a function called
     twoInARow
 that takes TWO arguments:
    -- an array of int's
    -- the length of the array
 The function RETURNS 1 ("true") if the array has two entries, one right after
 the other, that are the same number.  Otherwise, it returns 0 ("false").

 For example, if the array contains these 6 numbers:
   7  8  -18   12  12  102
 then the function returns 1 ("true") because 12 appears immediately after 12,
 while if the array contains these 6 numbers:
   7  8  -18   7  8  7
 then the function returns 0 ("false") because no number appears immediately
 after itself.
********************************************************************************/
```