2025-08-12

# One-Dimensional Arrays (Solutions)

## Multiple Choices

1. What is the correct way of creating an array of int of size 5 named myArray?
   * int[] myArray = new int[5];
   * int[] myArray = int[5];
   * int[5] myArray = new int[];
   * int[4] myArray = new int[];
   * int myArray = new int[5];
   * int[] myArray = new int[4];
   * int[] myArray = new int(5);
   * int[] myArray = int[4];
2. Consider the following code:

* int[] grades = {10, 20, 5, 15};  
  Console.WriteLine(grades[2]);
* What will it display?
  + 5
  + Nothing
  + 20
  + 15
  + grades
  + grades[2]
  + 10

1. In the following statement, the value 5 is called the…

* string[] names = new string[5];
  + allocation limit
  + size declarator
  + length
  + upper bound

1. Each element in an array is assigned a unique number called the…
   * subscript
   * index
   * position
   * rank
2. Consider the following code:

* char[] grades = {'A', 'B', 'C', 'D', 'F'};  
  int i = 0;  
  while(i < grades.Length){  
   i++;  
   Console.WriteLine(grades[i]);  
  }
* Something is wrong with it, can you tell what?
  + There will be an “Index was outside the bounds of the array.” error.
  + The array is not properly initialized.
  + The loop is infinite
  + grades.Length is not declared.

## Exercises

### Syntax

1. Write a statement that creates a 10-element int array named numbers.

* Solution
* int[] numbers = new int[10];

1. Write a statement that creates and initializes an array of double with the values 12.5, 89.0 and 3.24.

* Solution
* double[] question = {12.5, 89.0, 3.24};

1. In the following, what is the value of the size declarator? What is the value of the index?

* int[] numbers;  
  numbers = new int[8];  
  numbers[4] = 9;
* Solution
* The size declarator is 8, the subscript, or index, is 4.

1. What is “array bounds checking”? When does it happen?

* Solution
* It is when C# makes sure that you’re not using a subscript outside the allowed range. It happens at run time, so after the program is already compiled, when it is executed.

1. Is there an error with the following code? If you think there is one, explain it, otherwise draw the content of the myIncomes array once those statements have been executed.

* double[] myIncomes = new double[5];  
  myIncomes[1] = 3.5;  
  // No income on day two.  
  myIncomes[3] = 5.8;  
  myIncomes[4] = 0.5;  
  myIncomes[5] = 1.5;
* Solution
* The subscripts are off. They should go from 0 to 4.

1. What would be the size of the test array after the following statement has been executed?

* int[] test = {3, 5, 7, 0, 9};
* Solution
* 5

1. What is wrong with the following array declaration?

* int[] books = new int[-1];
* Solution
* The size declarator cannot be negative.

1. Suppose we have an array named temp that has been declared and initialized. How can we know the number of elements in this array?

* Solution
* By using the Length field: temp.Length is the number of elements in the temp array.

1. What is the value of count and the content of number once the following has been executed?

* int count=2;  
  int[] number={3, 5, 7};  
  number[count--] = 8;  
  number[count]--;
* Solution
* The value of count is 1, numbers contains the elements 3, 4, 8.

1. Describe what the following code would do.

* int[] record = { 3, 8, 11 };  
  int accumulator = 0;  
  foreach (int i in record)  
   accumulator += i;
* Solution
* This code declares and initializes an int array with the values 3, 8, and 11, and then sum those values in an accumulator variable.

### Displaying Arrays

1. Write code that displays the first and last element of an array.

* Solution
* int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
    
   Console.WriteLine($"First element: {numbers[0]}");  
   Console.WriteLine($"Last element: {numbers[numbers.Length - 1]}");

1. What will be displayed at the screen by the following code?

* int[] values = new int[6];  
  for (int i = 0 ; i < 6 ; i++)  
   values[i] = (i \* 2 );  
    
  foreach (int j in values)  
   Console.WriteLine(j);
* Solution
* 0 2 4 6 8 10

1. Suppose we are given an int array dailyPushUp with 7 elements. Write a piece of code that displays the value of the elements stored in the array dailyPushUp.

* Solution
* for (int j = 0; j < 7; j++)  
   Console.WriteLine(dailyPushUp[j]);

1. Write code that displays every element in a numbers array of integers.

* Solution
* for (int i = 0; i < numbers.Length; i++)  
   {  
   Console.WriteLine(numbers[i]);  
   }

1. Write code that displays every element in a numbers array of integers in reverse order.

* Solution
* for (int i = (numbers.Length - 1) ; i >= 0 ; i-- )  
   {  
   Console.WriteLine(numbers[i]);  
   }

1. Write code that displays every other element in a numbers array of integers (that is, every even index).

* Solution
* for (int i = 0; i < numbers.Length; i += 2)  
   {  
   Console.WriteLine(numbers[i]);  
   }

1. Given an array numbers and a variable x, write code that displays every value in myArray that is equal to or greater than x.

* Solution
* for (int i = 0; i < numbers.Length; i++)  
   {  
   if (numbers[i] >= x)  
   {  
   Console.WriteLine(numbers[i]);  
   }  
   }

1. Write code that displays every unique value of *a sorted* array numbers. This array could contain, for example, 1, 1, 1, 4, 4, 5, 8, 9, 11: values are increasing and can occur multiple times. In our example, the values 1, 4, 5, 8, 9 and 11.

* Solution
* for (int i = 0 ; i < numbers.Length ; i++)  
   {  
   if (i > 0 && numbers[i] != numbers[i - 1])  
   {  
   Console.WriteLine(numbers[i]);  
   }  
   }

### Filling Arrays

1. Given an array myArray and some value x, write code that sets the value of every array element to x.

* Solution
* for (int i = 0; i < myArray.Length; i++)  
   {  
   myArray[i] = x;  
   }

1. Given some positive number n, write code that first declares an array of length n, then sets its contents to sequentially increasing values 1, 2, 3, …, n.

* Solution
* int[] nums = new int[n];  
    
   for (int i = 0 ; i < n ; i++)  
   {  
   nums[i] = i + 1;  
   }

1. Given an array of integers, and two integer variables oldValue and newValue; write code that replaces every occurrence of oldValue in the array with newValue.

* Solution
* for (int i = 0; i < myArray.Length; i++)  
   {  
   if (myArray[i] == oldValue)  
   {  
   myArray[i] = newValue;  
   }  
   }

1. Write code that squares every value in an myArray integer array. For example, an array containing 2, 3, 4 would after the program contain 4, 9, 16.

* Solution
* for (int i = 0; i < myArray.Length; i++)  
   {  
   myArray[i] \*= myArray[i];  
   }

### Looking For Values

1. Given an array myArray and a value val, write code that checks if myArray contains val. The result should be true if val occur in myArray and false otherwise.

* Solution
* bool valOccurs = false;  
    
   for (int i = 0; i < myArray.Length; i++)  
   {  
   if (myArray[i] == val)  
   {  
   valOccurs = true;  
   }  
   }

1. Given an array myArray and a variable x, write code that computes the number of times x occurs in myArray.

* Solution
* int count = 0;  
   for (int i = 0; i < array.Length; i++)  
   {  
   if (x == array[i])  
   {  
   count++;  
   }  
   }

1. Given an array myArray and two values x and y, write code that checks if myArray contains *either* x or y. The result should be true if x or y occur in myArray and false otherwise.

* Solution
* bool eitherOccurs = false;  
    
   for (int i = 0; i < array.Length; i++)  
   {  
   if (array[i] == val1 || array[i] == val2)  
   {  
   eitherOccurs = true;  
   }  
   }

1. Given an array myArray and two values x and y, write code that checks if myArray contains *both* values x and y. The result should be true when both values occur and false otherwise.

* Solution
* bool xOccurs = false, yOccurs = false, bothOccur = false;  
    
   for (int i = 0; i < myArray.Length; i++)  
   {  
   if (myArray[i] == x)  
   {  
   xOccurs = true;  
   }  
    
   if (myArray[i] == y)  
   {   
   yOccurs = true;  
   }  
   }  
    
   if (xOccurs && yOccurs)  
   {  
   bothOccur = true;  
   }

1. Given an integer myArray and a strictly positive integer value x, find an array element whose value is largest while also being strictly less than x and display it, or display 0 if there is no such value. For example, in an array containing 1, 2, 6, 7, 3, 9 with x being 8, the solution is 7.

* Solution
* int largest = 0;  
   for (int i = 0; i < myArray.Length; i++)  
   {  
   if (myArray[i] < x && myArray[i] > largest)  
   {  
   largest = myArray[i];  
   }  
   }  
    
   Console.WriteLine(largest);

1. Consider an array of char. Implement code to check if the array values form a palindrome, i.e., it reads the same way forwards and backwards.

* Solution
* bool palindromeSoFar = true;  
   int n = myArray.Length;  
   for (int i = 0; i < (n / 2); i++)  
   {  
   if (myArray[i] != myArray[(n - 1) - i]) // The two sides of the word are not equal to each other.  
   {  
   palindromeSoFar = false;  
   }  
   }
* Note that after the code has been executed, both sides of the word have been checked, and palindromeSoFar is true if both sides are mirrors of each other.

### Manipulating Two Arrays

1. Assuming we have two int arrays of the same size, firstA and secondA, write a program that copies the content of firstA into secondA.

* Solution
* for (int k =0; k < firstA.Length; k++)  
   secondA[k] = firstA[k];

1. Given two arrays array1 and array2, write a program to determine if there exists a value that occurs in both arrays. If such value exists, the result should be true and false otherwise.

* Solution
* bool valueInCommon = false;  
    
   for (int i = 0; i < array1.Length; i++)  
   {  
   for (int j = 0; j < array2.Length; j++)  
   {  
   if (array1[i] == array2[j])  
   {  
   valueInCommon = true;  
   }  
   }  
   }

1. Write a program that combines two string arrays called array1 and array2 into a single array containing first all the elements from array1, then all the elements from array2.

* Solution
* string[] combined = new string[array1.Length + array2.Length];  
    
   for (int i = 0; i < array1.Length; i++)  
   {  
   combined[i] = array1[i];  
   }  
    
   for (int j = 0; j < array2.Length; j++)  
   {  
   combined[array1.Length + j] = array2[j];  
   }

1. Given two arrays arrayA and arrayB, write code to check if every element in arrayB occurs in arrayA, then display the result as true or false.

* Solution
* Using while loops:
* int i = 0;  
   int j;  
   bool containsB = true;  
   bool containCurrentVal = false;  
   while(i < arrayB.Length && containsB)  
   {  
   j = 0;  
   containCurrentVal = false;  
   while (j < arrayA.Length && !containCurrentVal)  
   {  
   if(arrayA[j] == arrayB[i])  
   {  
   containCurrentVal = true;  
   }  
   j++;  
   }  
   if (!containCurrentVal)  
   {  
   containsB = false;  
   }  
   else  
   {  
   containCurrentVal = false;  
   }  
   i++;  
   }
* Using break:
* bool containsB = false;  
    
   for (int i = 0; i < arrayB.Length; i++)  
   {  
   containsB = false;  
    
   for (int j = 0; j < arrayA.Length; j++)  
   {  
   if (arrayA[j] == arrayB[i])  
   {  
   containsB = true;  
   break;  
   }  
   }  
    
   if (!containsB)  
   {  
   break;  
   }  
   }

### Methods

1. Write a static method (header included) that takes as an argument an int array, and displays on the screen the value of each element of that array.

* Solution
* public static void displayArray(int[] arrayP){  
   foreach (int element in arrayP)  
   Console.WriteLine(element);  
   }

1. Write a static method (header included) that takes as an argument an int array, and stores the value 10 in each element of that array.

* Solution
* public static void fillArray(int[] arrayP){  
   for (int j = 0; j < arrayP.Length; j++)  
   arrayP[j] = 10;  
   }

## Simple Algorithms

1. Write a program that computes the sum of values stored in a numbers array of integers and displays it.

* Solution
* int sum = 0;  
    
   for (int i = 0; i < numbers.Length; i++)  
   {  
   sum += numbers[i];  
   }  
    
   Console.WriteLine($"The sum is {sum}.");

1. Given an array of positive integers, count how many even values occur in that array.

* Solution
* int count = 0;  
    
   for (int i = 0; i < myArray.Length; i++)   
   {  
   if (myArray[i] % 2 == 0)  
   {  
   count++;  
   }  
   }

1. Write a program that computes the average of the elements in a arrayP numeric array.

* Solution
* int sum = 0;  
    
   for (int i = 0; i < arrayP.Length; i++)  
   {  
   sum += arrayP[i];  
   }  
    
   double average = (double)sum / arrayP.Length;

1. Write a program that finds the largest value in an integer array arrayP.

* Solution
* int maxSoFar = arrayP[0];  
   foreach (int element in arrayP)  
   {  
   if (element > maxSoFar)  
   {  
   maxSoFar = element;  
   }  
   }

1. Write a program that finds the smallest value in an integer array arrayP.

* Solution
* int minSoFar = arrayP[0];  
   foreach (int element in arrayP)  
   {  
   if (element < minSoFar)  
   {  
   minSoFar = element;  
   }  
   }

1. Write a program that finds the *second* smallest value in an array of integers arrayP.

* Solution
* int smallest = arrayP[0];  
   int secondSmallest = arrayP[1];  
    
   if (smallest > secondSmallest)  
   {  
   int temp = smallest;  
   smallest = secondSmallest;  
   secondSmallest = temp;  
   }  
    
   for (int i = 2; i < arrayP.Length; i++)  
   {  
   if (arrayP[i] < smallest)  
   {  
   secondSmallest = smallest;  
   smallest = arrayP[i];  
   }  
   else if (arrayP[i] < secondSmallest && arrayP[i] > smallest)  
   {  
   secondSmallest = arrayP[i];  
   }  
   }

1. Write code that finds the index of the first occurrence of a value val in an array arrayP. If the array does not contain the value, the result should be -1.

* Solution
* int index = -1;  
    
   for (int i = arrayP.Length -1; i >= 0; i--)  
   {  
   if (arrayP[i] == val)  
   {  
   index = i;  
   }  
   }

1. Write code that finds the index of the last occurrence of a value in an array. If the array does not contain the value, the result should be -1.

* Solution
* int index = -1;  
    
   for (int i = 0; i < arrayP.Length; i++)  
   {  
   if (arrayP[i] == val)  
   {  
   index = i;  
   }  
   }

1. Write code to reverse the contents of an array myArray. For example, an array containing 1, 4, 3, 2, 5 should contain, after the program was executed, 5, 2, 3, 4, 1.

* Solution
* for (int i = 0, j = myArray.Length - 1; i < j; i++, j--)  
   {  
   int temp = myArray[i];  
   myArray[i] = myArray[j];  
   myArray[j] = temp;  
   }

## Problems on Binary Search

Those two problems are related to [binary search](https:/princomp.github.io/lectures/collections/search#binary-search).

1. Assume arrayEx is an array of int containing the following values:

* int[] arrayEx =  
   {  
   10,  
   15,  
   30,  
   45,  
   60,  
   75,  
   90,  
   105,  
   120,  
   135,  
   150,  
   165,  
   180,  
   };
* and consider the following algorithm:
* bool fsf = false;  
   int tar = 105;  
   int sta = 0;  
   int end = arrayEx.Length - 1;  
   int mid,  
   cur;  
   while (sta <= end && !fsf)  
   {  
   mid = (sta + end) / 2;  
   cur = arrayEx[mid];  
   if (tar == cur)  
   {  
   fsf = true;  
   }  
   else if (tar > cur)  
   {  
   sta = mid + 1;  
   }  
   else  
   {  
   end = mid - 1;  
   }  
   }
* Complete the following table, giving the value of sta, end, mid, fsf and cur before the while loop is executed, after it has executed one time, two times, etc. If the value is not set at this point, write “undef”, and if the loop stops before the Xth iteration, simply cross out the Xth iteration. Report the value even if it did not change.

|  | Before loop | After 1 iteration | After 2 iterations | After 3 iterations | After 4 iterations |
| --- | --- | --- | --- | --- | --- |
| sta |  |  |  |  |  |
| end |  |  |  |  |  |
| mid |  |  |  |  |  |
| fsf |  |  |  |  |  |
| cur |  |  |  |  |  |

* Solution
* The code can easily be edited to display the information we are looking for.
* fsf = false;  
   tar = 105;  
   sta = 0;  
   end = arrayEx.Length - 1;  
   string sep = new String('-', 72); // Simple separating string, for formatting purpose.  
   Console.WriteLine("Before loop:\n" + sep);  
   Console.WriteLine(  
   "| {0, 5} | {1, 5} | {2, 5} | {3, 5} | {4, 5} |",  
   $"sta = {sta}",  
   $"end = {end}",  
   $"mid = undefined",  
   $"fsf = {fsf}",  
   $"cur = undefined"  
   );  
   Console.WriteLine(sep);  
   sep = new String('-', 57); // Shortening the separating string.  
   int counter = 0; // To display loop count.  
   while (sta <= end && !fsf)  
   {  
   counter++;  
   mid = (sta + end) / 2;  
   cur = arrayEx[mid];  
   if (tar == cur)  
   {  
   fsf = true;  
   }  
   else if (tar > cur)  
   {  
   sta = mid + 1;  
   }  
   else  
   {  
   end = mid - 1;  
   }  
   Console.WriteLine(  
   $"After {counter} iteration(s):\n" + sep  
   );  
   Console.WriteLine(  
   "| {0, 5} | {1, 5} | {2, 5} | {3, 5} | {4, 5} |",  
   $"sta = {sta}",  
   $"end = {end}",  
   $"mid = {mid}",  
   $"fsf = {fsf}",  
   $"cur = {cur}"  
   );  
   Console.WriteLine(sep);  
   }
* [*(Download this code)*](code/projects/BinarySearch.zip)
* Before loop:  
  ------------------------------------------------------------------------  
  | sta = 0 | end = 12 | mid = undefined | fsf = False | cur = undefined |  
  ------------------------------------------------------------------------  
  After 1 iteration(s):  
  ---------------------------------------------------------  
  | sta = 7 | end = 12 | mid = 6 | fsf = False | cur = 90 |  
  ---------------------------------------------------------  
  After 2 iteration(s):  
  ---------------------------------------------------------  
  | sta = 7 | end = 8 | mid = 9 | fsf = False | cur = 135 |  
  ---------------------------------------------------------  
  After 3 iteration(s):  
  ---------------------------------------------------------  
  | sta = 7 | end = 8 | mid = 7 | fsf = True | cur = 105 |  
  ---------------------------------------------------------

1. Consider the following **incorrect** implementation of binary search:

* int[] arrayP = {10, 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165};  
  bool fsf = false; // Found target so far?  
  int tar = 165; // Target to find.  
  int sta = 0; // Beginning  
  int end = arrayP.Length - 1; // End  
  Console.WriteLine($"Starting ({sta}-{end})");  
  while (sta <= end && !fsf){ // Here goes the main loop  
   if (tar < arrayP[(sta + end) / 2]) { end = (sta + end) / 2;  
   Console.WriteLine($"Going left ({sta}-{end})"); }  
   else if (tar == arrayP[(sta + end) / 2]) fsf = true;  
   else { sta = (sta + end) / 2;  
   Console.WriteLine($"Going right ({sta}-{end})"); }  
  }  
  Console.WriteLine("Found: " + fsf);
  1. Indicate what would be displayed
  + Solution
  + It would display
  + Starting (0-11)  
    Going right (5-11)  
    Going right (8-11)  
    Going right (9-11)  
    Going right (10-11)  
    Going right (10-11)  
    Going right (10-11)
  + and loop forever.
  1. Explain why this implementation is incorrect, and how it can be fixed.
  + Solution
  + The problem is that mid will never reach arrayP.Length -1. Indeed, we will have an expression equivalent to ((arrayP.Length -2) - arrayP.Length-1) / 2 (for example, as above, (10 + 11) / 2) that will never be rounded up to arrayP.Length-1 (in our example, 11), but always truncated.
* To fix this implementation, instead of end = (sta + end) / 2;, we should have end = ((sta + end) / 2) + 1;.

## Wrap-Up Problems

1. Declare and initialize three arrays:
   * Choose different data type for each array,
   * Make the arrays have different lengths: 3, 5, 10 elements respectively,
   * Initialize each array with appropriate values of your choice (depends on the type).

* After you have declared and initialized the arrays, display on the screen
  + The first value from array 1 (0th index),
  + The last value from array 2 (4th index),
  + The first three values from array 3 (indexed 0 - 2).
* Example Solution
* // Initialize arrays  
   string[] names = { "Alice", Bob", "Charlie" };  
   float[] tenths = { 0.1f, 0.2f, 0.3f, 0.4f, 0.5f };  
   int[] evens = { 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 };  
    
   // Display elements from arrays  
   Console.WriteLine( names[0] ); // Displays: Alice  
   Console.WriteLine( tenths[4] ); // Displays: 0.5   
   Console.WriteLine( evens[0] + " " + evens[1] + " " + evens[2] ); // Displays: 2 4 6

1. Suppose given two 1-dimensional arrays of char called passcode and codeentered of the same size.
   1. Write a program that displays “Passcode correct” if the content of passcode and of codeentered are identical.
   2. Write a program that displays “Passcode valid” if passcode does not contain the same character twice.
   3. Write a program that displays “Passcode sorted” if passcode is sorted (either in ascending or descending order).
   4. Write a program that displays the number of matching characters in the same position in both arrays. For example, if passcode contains b, C, 4, a and codeentered contains B, 4, c, a, then “1” should be displayed, as only one character (a) is in both arrays at the same position.
   5. Write a program that displays the number of matching characters in both arrays. For example, if passcode and codeentered are as in the previous question, then “2” should be displayed, as two characters (4 and a) are in both arrays[[1]](#footnote-32).

* Solution
* A possible implementation is as follows:
* using System;  
    
  class Program  
  {  
   public static void Main()  
   {  
   char[] passcode = { 'b', 'C', '4', 'a' };  
   char[] codentered = { 'B', '4', 'c', 'a' };  
    
   // Test for identity  
   bool identical = true;  
   for (int i = 0; i < passcode.Length; i++)  
   {  
   if (passcode[i] != codentered[i])  
   identical = false;  
   }  
   if (identical)  
   {  
   Console.WriteLine("Passcode correct");  
   }  
   // Test for validity  
   bool valid = true;  
   for (int i = 0; i < passcode.Length; i++)  
   {  
   for (int j = i + 1; j < passcode.Length; j++)  
   if (passcode[i] == passcode[j])  
   valid = false;  
   }  
   if (valid)  
   {  
   Console.WriteLine("Passcode valid");  
   }  
   // Test for sorted  
   bool sorted\_a = true; // sorted ascending  
   bool sorted\_d = true; // sorted descending  
   for (int i = 0; i < passcode.Length - 1; i++)  
   {  
   if (passcode[i] > passcode[i + 1])  
   sorted\_a = false;  
   if (passcode[i] < passcode[i + 1])  
   sorted\_d = false;  
   }  
   if (sorted\_a || sorted\_d)  
   {  
   Console.WriteLine("Passcode sorted");  
   }  
   // Matching chars -- Same position  
   int count1 = 0;  
   for (int i = 0; i < passcode.Length; i++)  
   {  
   if (passcode[i] == codentered[i])  
   count1++;  
   }  
   Console.WriteLine(  
   "Number of matching characters (at the same position) : "  
   + count1  
   + "."  
   );  
   // Matching chars -- Any position  
   int count2 = 0;  
   for (int i = 0; i < passcode.Length; i++)  
   {  
   for (int j = 0; j < codentered.Length; j++)  
   if (passcode[i] == codentered[j])  
   count2++;  
   }  
   Console.WriteLine(  
   "Number of matching characters (at any position) : "  
   + count2  
   + "."  
   );  
   }  
  }
* [*(Download this code)*](https:/princomp.github.io/code/projects/Passcode.zip)
* It is recommended to test it with other values to make sure that the program behave as expected.

1. Consider this array of words:

* string[] words = { "voice",   
   "effect",  
   "day",  
   "orange",  
   "appliance",  
   "fly",  
   "cloud",  
   "degree",  
   "engine",  
   "society"};
* Write code to display an answer to the following questions. If the solution requires looping, use **foreach** loop when possible. Otherwise, use a **for** loop.
  1. Does words array contain “engine”? (true/false)
  2. Does words array contain “day” at least 2 times? (true/false)
  3. What is the position (index) of the word “society”? If it does not exist answer should be -1. Find the position of the first occurrence in case there are multiple matches.
* After you have implemented your code, change the array contents and make sure your code still works and does not crash.
* Here is another array of words to test your solution:
* string[] words = {"addition", "day", "start", "dock", "fowl", "fish", "seat", "day"};
* Solution
* // Code for finding the word "engine" in the array  
   bool foundEngine = false;  
    
   for (int i = 0; i < words.Length; i++)  
   {  
   if (words[i] == "engine")  
   {  
   foundEngine = true;  
   }  
   }  
    
   Console.WriteLine("Array contains \"engine\": " + foundEngine);  
    
   // Code for finding the word "day" in the array at least twice  
   bool foundDay2x = false;  
   int dayCount = 0;  
    
   for (int i = 0; i < words.Length; i++)   
   {  
   if (words[i] == "day")  
   {  
   dayCount++;  
   }  
   }  
    
   if (dayCount >= 2)   
   {  
   foundDay2x = true;  
   }  
    
   Console.WriteLine("Array contains \"day\" twice: " + foundDay2x);  
    
   // Code for finding the position of the word "society" in the array  
   int societyPos = -1;  
    
   for (int i = 0; i < words.Length; i++)  
   {  
   if (words[i] == "society")  
   {  
   societyPos = i;  
   break;  
   }  
   }  
    
   Console.WriteLine("Position of \"society\": " + societyPos);

1. Assume all the characters in passcode are different, and the same for codeentered: a, a, b, b would not be valid. [↑](#footnote-ref-32)