8-1: Arrays and Whole Numbers

Write a program that reads a series of a hundred test scores from a file. Each score is guaranteed to be a positive whole number value in the range 0 to 100. The program should track the number of instances of each value using an array. Once all values have been inputted, display the number of instances of each score from highest score to lowest score. If no instances of a particular score exist, the program should simply skip listing it.

Sample output:

```
Value      Occurrence

100              3

98               1

93               2

91               6

88               2

87               3

81               1

80               3

79               2

76               2

61               3

58               2

54               1

52               1
```

8-2: The Average Function

Write a program that includes an `average` function. The average function should take an array of integers and a number of elements as arguments. The function should return the average of the numbers in the array. The function should work for any size `int` array. Demonstrate your function in main.

8-3: Parallel Arrays

A local inventory system keeps track of where merchandise is located within backroom storage. Write a program that tracks data using parallel arrays.

Model #	Bin #
0123AV	A1
3938DD	B12
4384CD	D7
5162WD	B3
5673AX	C13
6783EW	A4
7892GW	B5

Write a program that provides two menu choices: Search by Model or Search by Bin. If the user selects Search by Model, input the model number and tell the user the bin in which the item is stored. If the user selects Search by Bin, input the bin number and tell the user the model number of the item stored there.

To assist you with the search process, create a generic `searchArray` function that uses a sequential search for a given string. The function should return the subscript of the element containing the data or -1 to indicate the data was not found.

8-4: Arrays on the Menu

The table of charges for a nearby toll road is below.

		END (Departure)			
		Eagle Way	I-234	Daltry Lane	Park Street
START	Eagle Way	0	1.33	1.56	2.28
	I-234	1.33	0	0.58	1.33
	Daltry Lane	1.56	0.58	0	0.95
	Park Street	2.28	1.33	0.95	0

Write a program that inputs the starting location (one of the on ramps listed on the left of the table) and the ending location (one of the exit locations listed across the top.) The program should output the trip cost found at the intersection of these two locations. Tip: It is up to you to determine how the user inputs starting point and ending point. However, it may be useful to provide the user with a menu (e.g., 1=Eagle Way, 2=I-234, and so on) to simplify input.