**1.** Write a function named "reverse" that takes as its arguments the following:

(1) an array of floating point values;

(2) an integer that tells how many floating point values are in the array.

The function must reverse the order of the values in the array. Thus, for example, if the array that's passed to

the function looks like this:

0 1 2 3 4

5.8 | 2.6 | 9.0 | 3.4 | 7.1

then when the function returns, the array will have been modified so that it looks like this:

0 1 2 3 4

7.1 | 3.4 | 9.0 | 2.6 | 5.8

The function should not return any value.

**2.** Write a function named "sum" that takes as its arguments the following:

(1) an array of floating point values;

(2) an integer that tells how many floating point values are in the array.

The function should return as its value the sum of the floating point values in the array. Thus, for example, if the

array that's passed to the function looks like this:

0 1 2 3 4

5.8 | 2.6 | 9.0 | 3.4 | 7.1

then the function should return the value 27.9 as its value.

**3.** Write a function named "location\_of\_largest" that takes as its arguments the following:

(1) an array of integer values;

(2) an integer that tells how many integer values are in the array.

The function should return as its value the subscript of the cell containing the largest of the values in the array.

Thus, for example, if the array that's passed to the function looks like this:

0 1 2 3 4

58 | 26 | 90 | 34 | 71

then the function should return the integer 2 as its value. If there is more than one cell containing the largest of

the values in the array, then the function should return the *smallest* of the subscripts of the cells containing the

largest values. For example, if the array that's passed to the function is

0 1 2 3 4 5 6

58 | 26 | 91 | 34 | 70 | 91 | 88

then the largest value occurs in cells 2 and 5 , so the function should return the integer value 2.

**4.** Write a function named "location\_of\_target" that takes as its arguments the following:

(1) an array of integer values;

(2) an integer that tells how many integer values are in the array;

(3) an integer "target value".

The function should determine whether the given target value occurs in any of the cells of the array, and if it

does, the function should return the subscript of the cell containing the target value. If more than one of the cells

contains the target value, then the function should return the largest subscript of the cells that contain the target

value. If the target value does not occur in any of the cells, then the function should return the sentinel value -1

. Thus, for example, if the target value that's passed to the function is 34 and the array that's passed to the

function looks like this:

0 1 2 3 4 5 6

58 | 26 | 91 | 34 | 70 | 34 | 88

then the target value occurs in cells 3 and 5 , so the function should return the integer value 5 .

**5.** Write a function named "rotate\_right" that takes as its arguments the following:

(1) an array of floating point values;

(2) an integer that tells the number of cells in the array;

The function should shift the contents of each cell one place to the right, except for the contents of the last cell,

which should be moved into the cell with subscript 0 . Thus, for example, if the array passed to the function

looks like this:

0 1 2 3 4

5.8 | 2.6 | 9.1 | 3.4 | 7.0

then when the function returns, the array will have been changed so that it looks like this:

0 1 2 3 4

7.0 | 5.8 | 2.6 | 9.1 | 3.4

The function should not return a value.

**6.** Write a function named "shift\_right" that takes as its arguments the following:

(1) an array of floating point values;

(2) an integer, call it "left", that tells the leftmost cell of the part of the array to be shifted;

(3) an integer, call it "right", that tells the rightmost cell of the part of the array to be shifted;

(4) a positive integer, call it "distance" that tells how many cells to shift by.

The function should make sure that left is less than or equal to right, and that distance is greater

than zero. If either of these conditions fails, the function should return the value 1 to indicate an error.

Otherwise it should shift by distance cells the contents of the array cells with subscripts running from

left to right . Thus, for example, if the array passed to the function looks like this:

0 1 2 3 4 5 6 7 8 9 10 ....

5.8 | 2.6 | 9.1 | 3.4 | 7.0 | 5.1 | 8.8 | 0.3 | -4.1 | 8.0 | 2.7 | etc.

and if left has the value 3 , right has the value 7 , and distance has the value 2 , then the

function should shift the contents of cells 3 , 4 , 5 , 6 , and 7 to the right by 2 cells, so that when the function

returns, the array will have been changed so that it looks like this:

0 1 2 3 4 5 6 7 8 9 10 ....

5.8 | 2.6 | 9.1 | ??? | ??? | 3.4 | 7.0 | 5.1 | 8.8 | 0.3 | 2.7 | etc.

The question marks in cells 3 and 4 indicate that we don't care what numbers are in those cells when the

function returns. Note that the contents of cells 8 and 9 have changed, but the contents of cell 10 is

unchanged. The function need not take any precautions against the possibility that the cells will be shifted

beyond the end of the array (the calling function should be careful not to let that happen).

**7.** Write a function named "subtotal" takes as its arguments the following:

(1) an array of floating point values;

(2) an integer that tells the number of cells in the array.

The function should replace the contents of each cell with the sum of the contents of all the cells in the original

array from the left end to the cell in question. Thus, for example, if the array passed to the function looks like

this:

0 1 2 3 4

5.8 | 2.6 | 9.1 | 3.4 | 7.0

then when the function returns, the array will have been changed so that it looks like this:

0 1 2 3 4

5.8 | 8.4 | 17.5 | 20.9 | 27.9

because 5.8 + 2.6 = 8.4 and 5.8 + 2.6 + 9.1 = 17.5 and so on. Note that the contents of cell 0 are not

changed. The function should not return a value.

**8.** Write a function named "concatenate" that copies the cells of one array into a larger array, and then

copies the cells of another array into the larger array just beyond the contents of the first array. The contents of

the cells will be integers. The arguments will be as follows:

(1) the first array that will be copied;

(2) the number of cells that will be copied from the first array;

(3) the second array that will be copied;

(4) the number of cells that will be copied from the second array;

(5) the large array into which all copying will be performed;

(6) the number of cells available in the large array.

If the function discovers that the number of cells in the large array is not large enough to hold all the numbers to

be copied into it, then the function should return 0 to indicate failure. Otherwise it should return 1 . The

function should not alter the contents of the first two arrays. To take an example, if the first two arrays passed

to the function look like this:

0 1 2 3 4 5 6 0 1 2 3

58 | 26 | 91 | 34 | 70 | 34 | 88 and 29 | 41 | 10 | 66

then, provided the size of the large array is at least 11, the large array should look like this when the function

returns:

0 1 2 3 4 5 6 7 8 9 10

58 | 26 | 91 | 34 | 70 | 34 | 88 | 29 | 41 | 10 | 66