

Emphasis on: One-Dimensional Arrays, Sorting, Input from a file

In the mathematical theory of sets, a set is defined as a collection of distinct (no two alike) items of the same type. In some programming languages, sets are a built-in data type. C++ does not have a built-in set type, though the Standard Template Library does provide a set type as an add-on to the language. We can simulate a set in any language using a one-dimensional array.

There are several operations that can be performed on sets. We will consider only three of them: union, intersection, and difference. These are binary operations requiring two sets as operands.

- The union of two sets A and B ($A + B$), is a set that contains elements that are in A or in B or in both.
- The intersection of two sets A and B ($A * B$) is a set that contains elements common to both A and B.
- The difference of two sets A and B ($A - B$) is a set that contains only the elements which are in A but not in B.

For example, if A and B are two sets of integers defined as
 $A = \{ 5, 7, 8, 10 \}$ and $B = \{ 3, 9, 10 \}$,
then their union ($A + B$) is the set $\{ 3, 5, 7, 8, 9, 10 \}$
their intersection ($A * B$) is the set $\{ 10 \}$
the difference of A and B ($A - B$) is the set $\{ 5, 7, 8 \}$
the difference of B and A ($B - A$) is the set $\{ 3, 9 \}$.

Input File

File **setdata.txt** contains an even number of lines of data. Each line represents the values to be assigned to a set. Each line contains up to 25 random positive integers with a space between consecutive integers. The integers are in no particular order, and a particular integer may appear more than once.

Example:

4 8 10 2 14 7 10 8

Processing

Write a program that repeats the following until end of file:

- 1) Read a line of data and store the input integers in set (array) A. Read another line of data and store the integers in set (array) B. Echo print both input sets.
- 2) Make sure that both arrays contain proper sets by eliminating duplicate values in each. Place set values in ascending order (while the order of set values is immaterial, the set is easier to read and more efficient to check if its values are in order, and it gives me an excuse to ask you to use a sort). You will probably find it easier to eliminate duplicates if you do the sort first.
- 3) Compute and print the union of A and B ($A + B$), the intersection of A and B ($A * B$), the difference of A and B ($A - B$), and the difference of B and A ($B - A$).

Example of the output format required:

A = { 3 1 2 4 }

B = { 8 4 6 5 3 6 4 }

{ 1 2 3 4 } + { 3 4 5 6 8 } = { 1 2 3 4 5 6 8 }

{ 1 2 3 4 } * { 3 4 5 6 8 } = { 3 4 }

{ 1 2 3 4 } - { 3 4 5 6 8 } = { 1 2 }

{ 3 4 5 6 8 } - { 1 2 3 4 } = { 5 6 8 }

Program Requirements:

- 1) Since the purpose of this assignment is to give you practice in array manipulations and in sorting and searching, you may not use the Standard Template set type or other predefined set functions.
- 2) You must call a sort function to sort each set. You may use either the bubble sort or the selection sort from the book to implement sorting. You could also use your bucket sort you developed for Assignment #2 as your sorting method.
- 3) You may assume that the data will be valid positive integers.
- 4) The set unions which you calculate should not contain any duplicate values. You might find it useful to have a function which eliminates duplicates. You could call it for your input sets as well as for the set union calculated.
- 5) The union, intersection, and difference operations should be implemented as functions to make it easier to use them in other applications where a set might be useful.
- 6) All union, intersection, and difference sets should be shown with values in ascending order (call your sort function again).
- 7) Sets should be output as shown in the examples (values printed with a space or a comma between consecutive values and enclosed in curly braces).

More examples of set operations:

Set A		Set B		Resulting Set
{ 1, 2, 3 }	+	{ 4, 5, 6 }	=	{ 1, 2, 3, 4, 5, 6 }
{ 1, 2, 3, 4 }	+	{ 2, 4, 6 }	=	{ 1, 2, 3, 4, 6 }
{ 1, 2, 3, 4 }	*	{ 2, 4, 6 }	=	{ 2, 4 }
{ 1, 2, 3 }	*	{ 4, 5, 6 }	=	{ }
{ 1, 2, 3, 4 }	-	{ 2, 4 }	=	{ 1, 3 }
{ 2, 4 }	-	{ 1, 2, 3, 4 }	=	{ }
{ 1, 2, 3, 4 }	-	{ 2, 4, 6 }	=	{ 1, 3 }
{ 2, 4, 6 }	-	{ 1, 2, 3, 4 }	=	{ 6 }
{ 1, 2, 3 }	-	{ 4, 5, 6 }	=	{ 1, 2, 3 }
{ 4, 5, 6 }	-	{ 1, 2, 3 }	=	{ 4, 5, 6 }
{ 1, 2, 3 }	-	{ 1, 2, 3, 4 }	=	{ }
{ 1, 2, 3, 4 }	-	{ 1, 2, 3 }	=	{ 4 }