COT 3100 Program #1 Spring 2014
Assigned: 1/18/2014
Check WebCourses for due date

Note: This program (and all programs for the course) are ONLY for students who signed up for the programming option. If you signed up for this option, you MUST DO ALL FIVE programs assigned.

General Program Directions (to be followed for all five programs)

Turn in a single source file, either C or Java, with the name designated in the note at the bottom that solves the problem described below. Please read all of your input from standard in and output to standard out. To test your program with input files, please pipe the input file into your program and pipe the output to another file. If you don't know how to do this, please see a TA to describe this process to you.

General Grading Criteria

Your submissions for all programs will be graded based on points in three categories:

1) Execution Points
2) Coding Style Points
3) Elements of Code Specific to the Problem
4) Following given restrictions, if any.

The first criteria will typically be 40% to 60% of the program grade. It will simply be based on the number of test cases your program produces correct output on before either finishing or crashing. (No test cases after a crash will be run to see if your program would have passed them.)

The second criteria will typically be 10% to 15% of the program grade. It will look very similar to my Computer Science I style criteria. Use good variable names, indent properly, use reasonable white space, and comment appropriately.

The third and fourth criteria will be worth the rest of each criteria. These will be based on elements of the code and following given restrictions. Occasionally points will be assigned for what is deemed to be a good programming practice, even if it's not necessary to solve the problem correct. (And you won't be told which strategy or data structure to use because I want to reward students who come up with these unique ideas on their own.)

The Problem: Set Operations

Write a program that calculates the basic set operations we learned for finite sets of integers. Namely, given two sets A and B of integers, determine the following sets:

$A \cup B$
$A \cap B$
$A - B$
AxB

Problem Solving Restrictions

You may use arrays in your solution, but you may not use any language construct that naturally takes care of any implementation issues with sets. For example, you are not allowed to use a TreeSet, since it naturally will only add an item if that item doesn't already appear in the TreeSet. If you have any questions about whether a particular class or API function is allowed, please ask me directly. If you don't, you run the risk of not adhering to this restriction.

Input Format

The first line of the input file will contain a single positive integer, *n*, representing the number of pairs of sets for which to calculate the four desired sets.

Each input case will follow with each case taking two lines. The first of these two lines will be the elements of set A, terminated with a -1, separated with spaces. The second of these two lines will list the elements of set B in the same format as the elements of set A were listed. You are guaranteed that each of these lists except the terminating -1 contain in between 1 and 20 distinct non-negative integers less than or equal to 100 that may not be in sorted order. Each of these lists will contain a maximum of 20 elements and none of the elements will exceed 100.

Output Format

For each test case, output a single header line with the format

```
Case k
```

where k is the number of the test case, starting with 1.

For each test case, output four lines, with the first line listing the elements of $A \cup B$ in sorted order with a space after each element, the second line listing the elements of $A \cap B$ in sorted order with a space after each element, the third line listing the elements of $A - B$ in sorted order with a space after each element, and the last line listing the elements of AxB in lexicographical order with a space after each element. To list each element of the Cartesian product, use a open parenthesis, followed by the first element, followed by comma, followed by the second element, followed by a close parenthesis. Lexicographical ordering means sort the pairs first by the first element and break ties using the second element.

Sample Input

```
2
3 8 2 9 1 -1
2 7 -1
1 6 -1
6 2 4 -1
```

Sample Output

```
Case 1
1 2 3 7 8 9
2
1 3 8 9
(1,2) (1,7) (2,2) (2,7) (3,2) (3,7) (8,2) (8,7) (9,2) (9,7)
Case 2
1 2 4 6
6
1
(1,2) (1,4) (1,6) (6,2) (6,4) (6,6)
```

Deliverables

A single source file, named either ***set.c*** or ***set.java*** that solves the program stated above, using the input and output formats stated above, using standard input and standard output. The file should be submitted via WebCourses by the due date and time stated in WebCourses.