Machine Problem 1: Ready get SET!

General Description

For this MP, you are to implement the set data structure.

The set data structure is a container that stores unique elements in no particular order which should allow for fast access. For this MP, implement the following operations for the set data structure:

1. insert
2. remove
3. subset
4. union
5. intersection
6. difference
7. power set

Implement the set as a template or generic that should allow the user to create different types of sets.

- set <int> s;
- set <string> s1;
- set < set<int> > s2;

For our purpose, we are to include int, double, char, string, set, and object.

Input File

mpa1.in not 2.

The input file (named mpa2.in) will contain a number of lines. The first line is the number of test cases. Each test case will contain a series of lines as well. The first line will contain a number representing what type of sets are to be created (exactly two sets will be created for each test case). The following are the different types:

1. int
2. double
3. char
4. string
5. set
6. object

If the type is of type set (5), then a second number will be found specifying what kind of a set its elements are. Take for instance the sample below:

```
1
5 1
```

This means that for this test case, the sets will contain sets of integers as elements.

The next line will contain the default elements (separated by a space) of the first set followed by another line containing the default elements (separated by a space) of the second set.

Another line with one number will follow. This number, say x, represents the number of operations that will be performed or executed for this particular test case.

x lines will complete the test case. Each line will contain the type of operation to be performed or executed followed by the necessary parameters for it to work. The following are the types of operations:

1. insert
2. remove
3. subset
4. union
5. intersection
6. difference
7. power set

For 1 (insert), two items will follow (separated by a space), a number (either 1 or 2) determining in which set the item is to be inserted and the item to be inserted. The same applies for 2 (remove).

For 3 (subset), no additional items will be found. This operation will simply verify whether the first set is a subset of the second.

The same goes for 4 (union), 5 (intersection), and 6 (difference).

- 4 : union of the two sets
- 5 : intersection of the two sets
- 6 : 1 - 2 (set difference)

For operation 7 (power set), a number will follow it (separated by a space) specifying for which set the operation is to be performed (1 or 2).

Below is a sample input file.

```
3                       Number of test cases
1                       Make an int set       (TestCase1)
1 5 7 8                 First set         (set1)
10 -2 4 5               Second set    (set2)
3                       Three operations
1 1 8                   1. Insert 8 at set1
1 2 7                   2. Insert 7 at set2
3                       3. Check if set1 is subset of set2
4                       Make a char set      (TestCase2)
hello abcd wxyz         First set         (set1)
world wxyz abcd lmno    Second set    (set2)
4                       Four operations
1 1 world               1. Insert world at set1
2 2 world               2. Remove world at set2
5                       3. Show intersection of the two sets
7 2                     4. Show all subsets of set2
5 1                     Make a set of sets    (TestCase3)
{1,2,3} {1,2} {7,8,9}   First set         (set1)
{1,2} {1,2,5} {7,8,9,10} Second set    (set2)
2                       Two Operations
2 2 {7,8,9,10}          1. Remove {7,8,9,10} at set2
7 2                     2. Show all subsets of set2
```

Output file

For each operation performed or executed, a line of output should be printed to a file named <lastname>1.out.

- for insert and remove, display the contents of the set affected
- for subset, simply print true or false, depending on the verification
- for union, intersection, difference and power set, the contents of the resulting set

Below is a sample output file of the sample input file above.

```
{1,5,7,8}
{10,-2,4,5,7}
false
{hello,abcd,wxyz,world}
{wxyz,abcd,lmno}
{abcd,wxyz}
{empty,{wxyz},{abcd},{lmno},{wxyz,abcd},{wxyz,lmno},{abcd,lmno},{wxyz,abcd,lmno}}
{{1,2},{1,2,5}}
{empty,{{1,2}},{{1,2,5}},{{1,2},{1,2,5}}}
```