**1. BitSet**

by Cherry Lyn Sta. Romana

A set of integers 0..MAX may be implemented using an array of 1/0 values. This particular implementation is called a bit-vector implementation of a Set.

For example, if the integer 3 is an element of the set, then the array element indexed by 3 is 1. On the other hand, if 3 is not an element, then the array element indexed by 3 is 0.

For example: if s = {3,4,6,8}, the array looks like this:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

Implement a programmer-defined data type called BitSet using dynamically allocated arrays. Implement the following functions:

```
BitSet newBitSet(); // allocate the array, contents of array should be 0
void add(BitSet s,int elem); // add an elem
void removeIt(BitSet s,int elem); // remove an element
void destroy(BitSet *s); // free the array and set pointer to NULL
void display(const char *name,const BitSet s); // this is given
void clear(BitSet s); // makes the set an empty set by setting all elements to 0

// return another BitSet containing the union of s1 and s2
BitSet getUnion(const BitSet s1,const BitSet s2);

// return another BitSet containing the intersection of s1 and s2
BitSet getIntersection(const BitSet s1,const BitSet s2);

// return another BitSet containing the difference of s1 and s2 (s1-s2)
// elements in s1 not in s2
BitSet getDifference(const BitSet s1,const BitSet s2);

// is elem an element of s? return 1 or 0
int isAnElement(const BitSet s,int elem);

// is the set empty? return 1 or 0
int isEmpty(const BitSet s);

// is s1 a subset of s2? return 1 or 0
int isSubset(const BitSet s1,const BitSet s2);

// are s1 and s2 disjoint?
int areDisjoint(const BitSet s1,const BitSet s2);
```

Input Format

first line contains the number of elements of the first set second line contains elements of the first set third line contains the number of elements of the second set fourth line contains elements of the second set the fifth line determines which functions will be tested 1: newBitset, add, removeIt, destroy 2. clear, isEmpty with 1 3. union with 1 and 2 4. intersection with 1,2,3 5. difference with 1,2,3,4 6. areDisjoint with 1,2,3,4,5 7. isSubset with 1,2,3,4,5,6

Input Sample

```
5
1·3·5·6·7
4
0·4·5·8
1
```

Output Format

s1 = { 1 2 5 6 9 } isEmpty(s1)? no s2 = { 0 3 4 8 } isEmpty(s2)? no union = { 0 1 2 3 4 5 6 8 9 } intersection = { } difference = { 1 2 5 6 9 } s1 = { } isEmpty(s1)? yes s2 = { } isEmpty(s2)? yes

Output Sample

```
s1·=·{·1·3·5·6·7·}
s2·=·{·0·4·5·8·}
s1·=·{·1·3·5·6·7·9·}
s2·=·{·4·5·8·}
```