Lab 2. Using C++, create three functions as pictured below. The first function should be a void function that calculates the union of two sets. The second function should be a void function that calculates the intersection of two sets. The third function should be a Boolean function that tells if the first set is a subset of the second or not.

Here is a possible variation: If you wish, you can instead create functions that return the result, as in (using my example)

```
      int_set set_union (int_set set1, int_set set2);
```
If you do that, you would then call the function like this:

```
      int_set A_union_B = set_union (A, B);
```
You can also do something similar for set_intersect

The ground rules are that you may not use any built-in set functions, except those that were used in the demo program. Specifically, what you <u>can</u> use are these:

Iterators and the begin() and end() functions, find(), erase(), insert(), size()

An example of the functions:

```
//  Create a set that can hold integers
   typedef set <int> int_set;
   int_set set_of_ints;

//  Display the elements of the set list_set
void list_elements (int_set list_set)

//  set_union (A, B, C) should set C to be the union of A and B
void set_union (int_set set1, int_set set2, int_set &result)

//  set_intersect (A, B, C) should set C to be the intersection of A
and B
void set_inter sect(int_set set1, int_set set2, int_set &result)

//  subset (A, B) is true if A is a subset of B and false otherwise
bool subset (int_set set1, int_set set2)
```