Set Theory

Assignment:
Create a program which is capable of assembling and displaying various sets I will provide you with. The program must correctly display the result of operators on sets (such as Union and Intersection), and **in addition it must do the following two things**: First, it must NOT repeat any values that might appear more than once. So if you Union two sets together that each have the number '5' somewhere in them, then '5' should only appear ONCE in the resulting set. Second, if the resulting thread is empty, your program must indicate the empty thread in some recognizable manner, such as displaying "Empty Thread" or "E" or something similar. Below are the threads you will use. Also assume that the A group comprises 'reality' so to speak, the numbers 1-10 being all elements possible, such that the INVERSE of A, is the empty thread. Thus if you were asked to compute the INVERSE of B, it would be equivalent to saying "Set A minus Set B". For this exercise, we will represent inverse with the ! symbol. *(100pts total: 60 for program, 5 per problem)*

A = {1,2,3,4,5,6,7,8,9,10}
B = {2,4,6,8,10}
C = {1,3,5,7,9}
D = {1,2,3,5,7}

Problem 1.) A intersect D
Problem 2.) (B union C) intersect A
Problem 3.) (!C union C) intersect A
Problem 4.) A – D
Problem 5.) **N**(!A union (C union D))
Problem 6.) D intersect C
Problem 7.) **N**(B intersect C)
Problem 8.) A union B union C union D

A reminder, N() indicates the MAGNITUDE of the resulting set. So if you have a set such as A = {1,3,7,10} then N(A) = 4. For problems asking for N(), your result should be a single number indicating the resulting set's size. Finally, a basic hint to help with this assignment: You may find arrays to be useful.

Code:

```
// Name: Dawlat Hamad
// ID: GV5450
// Lab 8 - Set Theory
// Source 1: https://cplusplus.com/reference/algorithm/set_union/
// Source 2: https://cplusplus.com/reference/algorithm/set_intersection/
// Source 3: https://cplusplus.com/reference/algorithm/set_difference/
// **You may be wondering how I knew which vector to use, the short answer is I spent a day
// **trying the different ones till I got correct results :)
```

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

int main ()
{
    //Declare Variables
    int A[ ] = {1,2,3,4,5,6,7,8,9,10};
    int B[ ] = {2,4,6,8,10};
    int C[ ] = {1,3,5,7,9};
    int D[ ] = {1,2,3,5,7};

    //Vector Functions
    vector<int> v1(10);
    vector<int>::iterator it1;
    vector<int> v2(10);
    vector<int>::iterator it2;
    vector<int> v3(10);
    vector<int>::iterator it3;
    vector<int> v4(10);
    vector<int>::iterator it4;
    vector<int> v5(10);
    vector<int>::iterator it5;
    vector<int> v6(10);
    vector<int>::iterator it6;
    vector<int> v7(10);
    vector<int>::iterator it7;

    //Sort Functions
    sort (A, A+10);
    sort (B, B+5);
    sort (C, C+5);
    sort (D, D+5);

    //Print Set Variables
    cout << endl;
    cout << "Set A: 1 2 3 4 5 6 7 8 9 10" << endl;
    cout << "Set B: 2 4 6 8 10" << endl;
    cout << "Set C: 1 3 5 7 9" << endl;
    cout << "Set D: 1 2 3 5 7" << endl;
    cout << endl;

    //Problem 1
    cout << "Problem 1.) A intersect D:";
```

```
it1 = set_intersection (A, A+10, D, D+5, v1.begin());
v1.resize(it1 - v1.begin());
for (it1 = v1.begin(); it1 != v1.end(); ++it1)
{
   if(*it1 == 0)
   {
      cout << " EMPTY SET";
   }
   else
   {
      cout << ' ' << *it1;
   }
}
cout << endl;

//Problem 2
cout << "Problem 2.) (B union C) intersect A:";
it2 = set_union (B, B+5, C, C+5, v2.begin());
v2.resize(it2 - v2.begin());
it3 = set_intersection (v2.begin(), v2.end(), A, A+10, v3.begin());
v3.resize(it3 - v3.begin());
for (it3 = v3.begin(); it3 != v3.end(); ++it3)
{
   if(*it3 == 0)
   {
      cout << " EMPTY SET";
   }
   else
   {
      cout << ' ' << *it3;
   }
}
cout << endl;

//Problem 3
cout << "Problem 3.) (!C union C) intersect A:";
it1 = set_difference (A, A+10, C, C+5, v1.begin());
v1.resize(it1 - v1.begin());
it2 = set_union (v1.begin(), v1.end(), C, C+5, v2.begin());
v2.resize(it2 - v2.begin());
it3 = set_intersection (v2.begin(), v2.end(), A, A+10, v3.begin());
v3.resize(it3 - v3.begin());
for (it3 = v3.begin(); it3 != v3.end(); ++it3)
{
   if(*it3 == 0)
   {
```

```
        cout << " EMPTY SET";
    }
    else
    {
        cout << ' ' << *it3;
    }
}
cout << endl;

//Problem 4
cout << "Problem 4.) A – D:";
it1 = set_difference (A, A+10, D, D+5, v1.begin());
v1.resize(it1 - v1.begin());
for (it1 = v1.begin(); it1 != v1.end(); ++it1)
{
    if(*it1 == 0)
    {
        cout << " EMPTY SET";
    }
    else
    {
        cout << ' ' << *it1;
    }
}
cout << endl;

//Problem 5
cout << "Problem 5.) N(!A union (C union D)):";
it2 = set_union (C, C+5, D, D+5, v2.begin());
v2.resize(it2 - v2.begin());
it3 = set_difference (A, A+10, A, A+10, v3.begin());
v3.resize(it3 - v3.begin());
it4 = set_union (v3.begin(), v3.end(), v2.begin(), v2.end(), v4.begin());
v4.resize(it4 - v4.begin());
if(v4.size() == 0)
{
    cout << " EMPTY SET";
}
else
{
    cout << " " << (v4.size());
}
cout << endl;

//Problem 6
cout << "Problem 6.) D intersect C:";
```

```
it1 = set_intersection (C, C+5, D, D+5, v1.begin());
v1.resize(it1 - v1.begin());
for (it1 = v1.begin(); it1 != v1.end(); ++it1)
{
   if(*it1 == 0)
   {
      cout << " EMPTY SET";
   }
   else
   {
      cout << ' ' << *it1;
   }
}
cout << endl;

//Problem 7
cout << "Problem 7.) N(B intersect C):";
it2 = set_intersection (C, C+5, B, B+5, v2.begin());
v2.resize(it2 - v2.begin());
if(v2.size() == 0)
{
   cout << " EMPTY SET";
}
else
{
   cout << " " << (v2.size());
}
cout << endl;

//Problem 8
cout << "Problem 8.) A union B union C union D:";
it7 = set_union (A, A+10, B, B+5, v7.begin());
v7.resize(it7 - v7.begin());
it5 = set_union (v7.begin(), v7.end(), C, C+5, v5.begin());
v5.resize(it5 - v5.begin());
it6 = set_union (v5.begin(), v5.end(), D, D+5, v6.begin());
v6.resize(it6 - v6.begin());
for (it6 = v6.begin(); it6 != v6.end(); ++it6)
{
   if(*it6 == 0)
   {
      cout << " EMPTY SET";
   }
   else
   {
      cout << " " << *it6;
```

```
        }
    }
    cout << endl;

    //END
    cout << endl;
    return 0;
}
```

Output:

```
Set A: 1 2 3 4 5 6 7 8 9 10
Set B: 2 4 6 8 10
Set C: 1 3 5 7 9
Set D: 1 2 3 5 7

Problem 1.) A intersect D: 1 2 3 5 7
Problem 2.) (B union C) intersect A: 1 2 3 4 5 6 7 8 9 10
Problem 3.) (!C union C) intersect A: 1 2 3 4 5 6 7 8 9 10
Problem 4.) A - D: 4 6 8 9 10
Problem 5.) N(!A union (C union D)): 6
Problem 6.) D intersect C: 1 3 5 7
Problem 7.) N(B intersect C): EMPTY SET
Problem 8.) A union B union C union D: 1 2 3 4 5 6 7 8 9 10
```