```cpp
/*  Program: A4P1 - IntSet
    Author: Tom Stutler
    Last Date Modified: 4/9/15

    The intent of this program to provide the user with an array of integer sets
    (0-9) and several operations for the sets to interact.
*/

#include <iostream>

using namespace std;

const int NUM_OF_INTS = 10, NUM_OF_SETS = 6; //Constants for array size.

class IntSet
{
public:
    IntSet() : intArray() {}
    ///Initializes all values in the set to false.

    friend ostream& operator <<(ostream& outputStream, IntSet& setParam);
    ///Displays the integers in the set in roster form, i.e. {1,3,5}.

    const IntSet operator +(IntSet& setParam);
    ///Returns the union of two sets.
    ///The union of sets A and B is the set that contains
    ///elements of set A or set B or both.
    const IntSet operator *(IntSet& setParam);
    ///Returns the intersection of two sets.
    ///The intersection of sets A and B is the set that
    ///contains all elements in both set A and B.
    const IntSet operator -(IntSet& setParam);
    ///Returns the difference of two sets.
    ///The difference of sets A and B is the set containing
    ///those elements that are in A but not B.
    const IntSet operator !();
    ///Returns the complement of a set.
    ///The complement of set A is the containing all the integers
    ///(0-9) that are not in set A.
    bool operator ==(IntSet& setParam);
    ///Returns true if set A is equal to set B and false if not.
    bool operator <=(IntSet& setParam);
    ///Returns true if set A is a subset of set B and false if not.
    void operator +=(int intParam);
    ///Adds an integer into the set.
    void operator -=(int intParam);
    ///Removes an integer from the set.

private:
    bool intArray[NUM_OF_INTS];
};

int selectset();
///Prompts the user to select which set to use and returns
///the integer value associated with it.

int displaymenu();
///Displays the menu to the user then prompts for and
///returns the user's selection.

int main()
{
    IntSet setArray[NUM_OF_SETS];
    int userChoice, enteredInt, currentSet, firstSet, secondSet;
    char repeat;

    do
    {
        userChoice = displaymenu();

        switch (userChoice)
        {
        case 1:
            cout << "Add numbers to which ";
            currentSet = selectset();

            do
            {
                cout << "Enter number to add: ";
                cin >> enteredInt;
                setArray[currentSet] += enteredInt;
                do
                {
                    cout << "Add another (y or n): ";
                    cin >> repeat;
                } while (repeat!='y' && repeat!='n' && repeat!='Y' && repeat!='N');
            } while (repeat=='y' || repeat=='Y');
            break;

        case 2:
            cout << "Remove numbers from which ";
            currentSet = selectset();

            do
            {
                cout << "Enter number to remove: ";
                cin >> enteredInt;
                setArray[currentSet] -= enteredInt;
                do
                {
                    cout << "Remove another (y or n): ";
                    cin >> repeat;
                } while (repeat!='y' && repeat!='n' && repeat!='Y' && repeat!='N');
            } while (repeat=='y' || repeat=='Y');
            break;

        case 3:
            cout << "Set union - specify sets to use:\n"
                 << "First ";
            firstSet = selectset();
            cout << "Second ";
            secondSet = selectset();
            cout << "Result ";
            currentSet = selectset();
            setArray[currentSet] = setArray[firstSet]+setArray[secondSet];
            break;

        case 4:
            cout << "Set intersection - specify sets to use:\n"
                 << "First ";
            firstSet = selectset();
            cout << "Second ";
            secondSet = selectset();
            cout << "Result ";
            currentSet = selectset();
            setArray[currentSet] = setArray[firstSet]*setArray[secondSet];
            break;

        case 5:
            cout << "Set difference - specify sets to use:\n"
                 << "First ";
            firstSet = selectset();
            cout << "Second ";
            secondSet = selectset();
            cout << "Result ";
            currentSet = selectset();
            setArray[currentSet] = setArray[firstSet]-setArray[secondSet];
            break;

        case 6:
            cout << "Set equality - specify sets to compare:\n"
                 << "First ";
            firstSet = selectset();
            cout << "Second ";
            secondSet = selectset();
            if (setArray[firstSet]==setArray[secondSet]) {
                cout << "These sets are equal.\n";
            } else {
                cout << "These sets are not equal.\n";
            }
            break;

        case 7:
            cout << "Set complement - specify sets to use:\n"
                 << "Complement ";
            firstSet = selectset();
            cout << "Result ";
            currentSet = selectset();
            setArray[currentSet] = !setArray[firstSet];
            break;

        case 8:
            cout << "Subsets - specify sets to compare:\n"
                 << "First ";
            firstSet = selectset();
            cout << "Second ";
            secondSet = selectset();
            if (setArray[firstSet]<=setArray[secondSet]) {
                cout << "The first set is a subset of the second.\n";
            } else {
                cout << "The first set is not a subset of the second.\n";
            }
            break;

        case 9:
            cout << "Display ";
            currentSet = selectset();
            cout << setArray[currentSet] << endl;
            break;

        case 0:
            return 0;
            break;
        }
    } while (userChoice != 0);
}

int selectset()
{
    int iset;
    char set;

    do
    {
        cout << "set (A,B,C,D,E,F)? :";
        cin >> set;
        set = toupper(set);
        iset = set-'A';
```

```cpp
        if (iset<0 || iset>5) {
            cout << "Invalid - reenter\n";
        }
    } while (iset<0 || iset>5);

    return iset;
}

int displaymenu()
{
    int selection;

    do
    {
        cout << "\nSelect an option:\n"
             << "1 - add numbers to a set\n"
             << "2 - remove numbers from a set\n"
             << "3 - form the union of two sets\n"
             << "4 - form the intersection of two sets\n"
             << "5 - form the difference of two sets\n"
             << "6 - determine if two sets are equal\n"
             << "7 - form the complement of a set\n"
             << "8 - determine if one set is a subset of another set\n"
             << "9 - display a set\n"
             << "0 - EXIT\n";

        cin >> selection;

        if (selection<0 || selection>9) {
            cout << "Invalid menu selection.\n";
        }

    } while (selection<0 || selection>9);

    return selection;
}

ostream& operator <<(ostream& outputStream, IntSet& setParam)
{
    outputStream << '{';
    for (int i=0; i<NUM_OF_INTS; i++) {
        if (setParam.intArray[i]==true) {
            outputStream << i;
            if (setParam.intArray[i+1]==true && i<NUM_OF_INTS-1) {
                outputStream << ", ";
            }
        }
    }
    outputStream << '}';

    return outputStream;
}

const IntSet IntSet::operator +(IntSet& setParam)
{
    IntSet unionSet;

    for (int i=0; i<NUM_OF_INTS; i++) {
        if (intArray[i] == true) {
            unionSet.intArray[i] = true;
        }
        if (setParam.intArray[i] == true) {
            unionSet.intArray[i] = true;
        }
    }

    return unionSet;
}

const IntSet IntSet::operator *(IntSet& setParam)
{
    IntSet intersectSet;

    for (int i=0; i<NUM_OF_INTS; i++) {
        if (intArray[i]==true && setParam.intArray[i]==true) {
            intersectSet.intArray[i] = true;
        }
    }

    return intersectSet;
}

const IntSet IntSet::operator -(IntSet& setParam)
{
    IntSet differenceSet;

    for (int i=0; i<NUM_OF_INTS; i++) {
        if (intArray[i]==true && setParam.intArray[i]==false) {
            differenceSet.intArray[i] = true;
        }
    }

    return differenceSet;
}

const IntSet IntSet::operator !()
{
    IntSet complimentSet;

    for (int i=0; i<NUM_OF_INTS; i++) {
        if (intArray[i]==false) {
            complimentSet.intArray[i] = true;
        }
    }

    return complimentSet;
}

bool IntSet::operator ==(IntSet& setParam)
{
    bool isEqual;

    for (int i=0; i<NUM_OF_INTS; i++) {
        if (intArray[i]==setParam.intArray[i]) {
            isEqual = true;
        } else {
            return false;
        }
    }

    return isEqual;
}

bool IntSet::operator <=(IntSet& setParam)
{
    bool isSubSet;

    for (int i=0; i<NUM_OF_INTS; i++) {
        if (intArray[i] == true) {
            if (setParam.intArray[i]==true) {
                isSubSet = true;
            } else {
```

```cpp
                    return false;
                }
            }
        }

        return isSubSet;
}

void IntSet::operator +=(int intParam)
{
    if (intParam>=0 && intParam<=9) {
        if (intArray[intParam] == false) {
            intArray[intParam] = true;
        } else {
            cout << intParam << " is already in that set.\n";
        }
    } else {
        cout << "Invalid value to add: " << intParam << endl;
    }
}

void IntSet::operator -=(int intParam)
{
    if (intParam>=0 && intParam<=9) {
        if (intArray[intParam] == true) {
            intArray[intParam] = false;
        } else {
            cout << intParam << " is not in that set.\n";
        }
    } else {
        cout << "Invalid value to add: " << intParam << endl;
    }
}
```