```
/*  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;

```

```
67      do
68      {
69          userChoice = displaymenu();
70
71          switch (userChoice)
72          {
73          case 1:
74              cout << "Add numbers to which ";
75              currentSet = selectset();
76
77              do
78              {
79                  cout << "Enter number to add: ";
80                  cin >> enteredInt;
81                  setArray[currentSet] += enteredInt;
82                  do
83                  {
84                      cout << "Add another (y or n): ";
85                      cin >> repeat;
86                  } while (repeat!='y' && repeat!='n' && repeat!='Y' && repeat!='N');
87              } while (repeat=='y' || repeat=='Y');
88              break;
89
90          case 2:
91              cout << "Remove numbers from which ";
92              currentSet = selectset();
93
94              do
95              {
96                  cout << "Enter number to remove: ";
97                  cin >> enteredInt;
98                  setArray[currentSet] -= enteredInt;
99                  do
100                 {
101                     cout << "Remove another (y or n): ";
102                     cin >> repeat;
103                 } while (repeat!='y' && repeat!='n' && repeat!='Y' && repeat!='N');
104             } while (repeat=='y' || repeat=='Y');
105             break;
106
107         case 3:
108             cout << "Set union - specify sets to use:\n"
109                  << "First ";
110             firstSet = selectset();
111             cout << "Second ";
112             secondSet = selectset();
113             cout << "Result ";
114             currentSet = selectset();
115             setArray[currentSet] = setArray[firstSet]+setArray[secondSet];
116             break;
117
118         case 4:
119             cout << "Set intersection - specify sets to use:\n"
120                  << "First ";
121             firstSet = selectset();
122             cout << "Second ";
123             secondSet = selectset();
124             cout << "Result ";
125             currentSet = selectset();
126             setArray[currentSet] = setArray[firstSet]*setArray[secondSet];
127             break;
128
129         case 5:
130             cout << "Set difference - specify sets to use:\n"
131                  << "First ";
132             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';
```

```
199         if (iset<0 || iset>5) {
200             cout << "Invalid - reenter\n";
201         }
202     } while (iset<0 || iset>5);
203 
204     return iset;
205 }
206 
207 int displaymenu()
208 {
209     int selection;
210 
211     do
212     {
213         cout << "\nSelect an option:\n"
214              << "1 - add numbers to a set\n"
215              << "2 - remove numbers from a set\n"
216              << "3 - form the union of two sets\n"
217              << "4 - form the intersection of two sets\n"
218              << "5 - form the difference of two sets\n"
219              << "6 - determine if two sets are equal\n"
220              << "7 - form the complement of a set\n"
221              << "8 - determine if one set is a subset of another set\n"
222              << "9 - display a set\n"
223              << "0 - EXIT\n";
224 
225         cin >> selection;
226 
227         if (selection<0 || selection>9) {
228             cout << "Invalid menu selection.\n";
229         }
230 
231     } while (selection<0 || selection>9);
232 
233     return selection;
234 }
235 
236 ostream& operator <<(ostream& outputStream, IntSet& setParam)
237 {
238     outputStream << '{';
239     for (int i=0; i<NUM_OF_INTS; i++) {
240         if (setParam.intArray[i]==true) {
241             outputStream << i;
242             if (setParam.intArray[i+1]==true && i<NUM_OF_INTS-1) {
243                 outputStream << ", ";
244             }
245         }
246     }
247     outputStream << '}';
248 
249     return outputStream;
250 }
251 
252 const IntSet IntSet::operator +(IntSet& setParam)
253 {
254     IntSet unionSet;
255 
256     for (int i=0; i<NUM_OF_INTS; i++) {
257         if (intArray[i] == true) {
258             unionSet.intArray[i] = true;
259         }
260         if (setParam.intArray[i] == true) {
261             unionSet.intArray[i] = true;
262         }
263     }
264 
```

```
265     return unionSet;
266 }
267
268 const IntSet IntSet::operator *(IntSet& setParam)
269 {
270     IntSet intersectSet;
271
272     for (int i=0; i<NUM_OF_INTS; i++) {
273         if (intArray[i]==true && setParam.intArray[i]==true) {
274             intersectSet.intArray[i] = true;
275         }
276     }
277
278     return intersectSet;
279 }
280
281 const IntSet IntSet::operator -(IntSet& setParam)
282 {
283     IntSet differenceSet;
284
285     for (int i=0; i<NUM_OF_INTS; i++) {
286         if (intArray[i]==true && setParam.intArray[i]==false) {
287             differenceSet.intArray[i] = true;
288         }
289     }
290
291     return differenceSet;
292 }
293
294 const IntSet IntSet::operator !()
295 {
296     IntSet complimentSet;
297
298     for (int i=0; i<NUM_OF_INTS; i++) {
299         if (intArray[i]==false) {
300             complimentSet.intArray[i] = true;
301         }
302     }
303
304     return complimentSet;
305 }
306
307 bool IntSet::operator ==(IntSet& setParam)
308 {
309     bool isEqual;
310
311     for (int i=0; i<NUM_OF_INTS; i++) {
312         if (intArray[i]==setParam.intArray[i]) {
313             isEqual = true;
314         } else {
315             return false;
316         }
317     }
318
319     return isEqual;
320 }
321
322 bool IntSet::operator <=(IntSet& setParam)
323 {
324     bool isSubSet;
325
326     for (int i=0; i<NUM_OF_INTS; i++) {
327         if (intArray[i] == true) {
328             if (setParam.intArray[i]==true) {
329                 isSubSet = true;
330             } else {
```

```
                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;
    }
}
```