CISC2000: Computer Science II
Set Class 3 - Operator Overloading

Objective: To *design and develop* a class in the C++ programming language that shows correct use of overloaded operators and dynamic memory.

Assignment: Define a class in C++ whose purpose it is to represent a Set of integers. All objects that are instantiated from this class represent subsets of the set of integers.

The class should be designed around the following characteristics:

1. The name of the class should be **Set.**

2. The class should be composed of **at least** the following (data) **members**:

   - ✓ **set**, pointer to a dynamically allocated array of integers. This array will contain the elements that comprise the set.
   - ✓ **pSize**, data member to represent the physical size of the array.
   - ✓ **numElements**, data member to represent the number of elements in the set.
   - ✓ **DEFAULT_SIZE**, (static constant) data member to *represent the default value used to dynamically allocate memory. Set the default size to five.*

   Note that you may need to add additional data members to accomplish the objective of the driver program. Also, make sure that the data type of each member is appropriate to the type of data being stored.

2. The class should include **at least** the following **methods**:
   - ✓ Set(int = DEFAULT_SIZE)
     Default constructor to initialize the set to the empty set. In other words, a set that contains no elements. *Note that this constructor should dynamically allocate as much memory as specified*. If a specific size is not passed, the DEFAULT_SIZE should be used.
   - ✓ Set(int [], int size)
     Constructor to initialize the set from the array of integers passed to the constructor. The arguments to this constructor are the array of integers, size of the passed array.
   - ✓ Set(Set)
     Set (Set&obj) Copy constructor to initialize the object from an existing object.
   - ✓ Destructor, clear out data members as necessary and deallocate all dynamically allocated memory as necessary.

- A method to populate the set of integers from user input. This method can prompt the user for the number of elements they want to enter into the set. If the number of elements they want to enter exceeds the physical size of the array, the array needs to be extended by an amount equal to DEFAULT_SIZE . You can dynamically create a new array, then copy the old array elements to the new array then deallocate the old array.
- ✓ A method to display the set. Note that the set should be displayed using proper set notation (i.e. { 1, 5, 7 }).
- Operator overloaded methods for the input and display method respectively.
- Mutator methods for each data member or logical groupings, as necessary.
- ✓ Methods to perform each of the following set operations:
  - ✓ union(Set)
    This method creates a new set that represents the union of the two sets (i.e. the set of the object this methods was invoked on with the set of the object passed to the method). This method returns the new set.
  - ✓ intersection(Set)
    This method creates a new set that represents the intersection of the two sets and returns the new set.
  - difference(Set)
    This method creates a new set that represetns the difference of the two sets and returns the new set.
  - equality(Set)
    This method determines whether the two sets are equal and return true or false accordingly.
  - inequalty(Set)
    This method determines whether the two sets are not equal and returns true or false accordingly.
  - Operator overloaded methods for each of the above set operations as follows:
    - ✓ **+** to represent the union of two sets.
    - ✓ **^** to represent the intersection of two sets.
    - **-** to represent the difference of two sets.
    - **==** to represent the equality of two sets.
    - **!=** to represent the inequality of two sets.

    Note the set operations {+,^,-} each result in a new set being created. Think carefully how you can determine the memory requirements of the array of the new set.

  - ✓ element(int)
    This method determines if the specified integer is an element of the set and returns true or false accordingly.

- ✓ A method to add a new element to an existing set and return {true, false} accordingly. *Note that by definition an element of a set cannot be repeated, therefore, this method must first ensure that the element being added is not already an element of the set. If the element cannot be added, the method should return false to denote this.* ***Further note that this method should be able to add a new element to the set even if the physical array is at capacity (i.e. number of elements equals the physical size) when the method is invoked.***

  If the number of elements they want to enter exceeds the physical size of the array, the array needs to be extended by an amount equal to DEFAULT_SIZE . You can dynamically create a new array, then copy the old array elements to the new array then deallocate the old array.

- A method to remove an element from an existing set and return {true, false} accordingly. *Note that for an element to be removed, it must be an element of the set.*
- ✓ Operator overloaded methods for the *add* and *remove* method as follows:
  - ✓ **+** to represent addition of a new element to the set.
  - **-** to represent deletion of an element from the set.

Note you can choose to directly implement each of the specified operator overloaded methods ***or*** *implement the methods {input, display, union, intersection, difference, equality, inequality, add, remove} as* ***private*** *methods which are invoked by their respective overloaded method.*

3. The class should allow for the instantiation of constant objects. In other words applications which use this class should be able to create sets as const.

4. ✓ Write a simple main function to instantiate several set objects for the purpose of testing out the various methods of the class. This main function can be defined in the class source file, but make sure it is commented (or removed) after the class has been tested.

   Create a set by adding 1, 2, 3, 4, 5, 1, 3, 5, 6 and print to get {1,2,3,4,5,6} order not important.

Important

1. .cpp files must only include .h or .hpp files and may not include other .cpp files. Also .h files may include other .h or .hpp files but not .cpp files.
2. Follow an incremental approach in developing the class. Compile and test the class as you implement each method. Do not wait until all the methods have been implemented before you begin testing.
3. If the methods compile but do not work as expected, consider using cout statements to display the contents of the appropriate members as well as to trace the flow of execution.

For full credit be sure to:

- Include a descriptive comment block at the beginning of the file
- Include a descriptive comment block before the defintion of each method.
- Use correct indentation and allignment.
- Use descriptive identifiers for variable names as well as appropriate data types.
- Use blank lines to separate the code into appropriate blocks.
- Include comments to help others understand your program, and help yourself think!