**Task#1**: Create a ‘**Set**’ ADT, which will implement the basic operations of Set concept in mathematics. The data members and operations needed for this ADT are given below.

**Data Members:**

* int \* data
* int noOfElements
* int capacity

**Supported Operations:**

The class '**Set**' should support following operations

1. **Set (int cap = 0);** Default parameterized constructor. Sets ‘cap’ to ‘capacity’ and initializes rest of the data members accordingly. If user sends any invalid value then sets the cap to default value.

1. **Set(const Set & ref);** overloaded copy constructor to implement deep copy.

# 3. ~Set()

Free the dynamically allocated memory.

1. **void insert (int element);** Stores the element in the Set.

**Remember!** Set is a collection of distinct elements. If the array is full, then resize the capacity of Set by double.

1. **void remove (int element);** remove the element from the Set. If the noOfElements in the Set are equal to or less than 50% of capacity then shrink the Set by 25% i.e., if the capacity was 12 and noOfElements are less than equals to 6 then the new Set should be of the capacity equals to 9.

1. **int getCardinality() const;** returns the number of elements in the set.

1. **Set calcUnion (const Set & s2 ) const;** returns a new Set object which contains the union of ‘s2’ set and calling object set.

1. **Set calcSymmetricDifference (const Set & s2);** returns a new Set object which contains the symmetric Difference of ‘s2’ set and calling object set. Where symmetric difference is: AΔB= AB−A∩B

1. **Set calcDifference (const Set & s2 ) const;** returns a new Set object which contains the intersection of ‘s2’ set and calling object set.

1. **int isMember (int val ) const;** returns 1 if ‘val’ is member of the set otherwise return 0.

1. **int isSubSet (const Set & s2) const;** returns 1 if s2 is proper subset of calling object set, return 2 if improper subset otherwise returns 0.

1. **void reSize (int newcapacity);** resize the set to new capacity. Make sure that elements in old set should be preserved in the new set if possible.

# 13. void update(int prVal, int curVal);

This function should replace the previous value(prVal) with current value (curVal). Make sure the property of set should not be violated (**distinct elements**). If the prVal is not present in the array, then display a message “**Target value not found**”. In-case of successful modification then display a message “**Record update successfully**”. If prVal is present in the set but curVal is also already available, then display a message “Violation set property. Can’t Modify the data”.

# 14. void show() const;

This function should display the contents of Set for calling object.

**Write main function to test the functionalities of supporting functions of Set ADT**

**Task#2:**

Create a class fraction with the following data members and supporting member functions.

**Data Members**

* static int fractionsCount;
* int numerator;
* int denominator;

**Member functions:**

1. **fraction(int n = 1, int d = 1);** default parameterized constructor assigns the value of ‘n’ to **numerator** and ‘d’ to **denominator**. The value of ‘d’ should not be zero. Increment the value of static data member.

1. **void setNum(int n);**

Setter function to assign the value of ‘n’ to numerator.

# 3. void setDen(int d);

Setter function to assign the value of ‘d’ to denominator. Value should not be zero.

1. **int getNum() const;** const getter function to return numerator

1. **int getDen() const;** const getter function to return denominator

1. **fraction operator + (const fraction & f);** overload addition operator (+). This function should store the sum of calling object and the parameter **f** in a newly created object and return the simplest form of object to the calling function. e.g.,

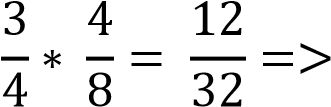
 4 6 32 42 74

+ = + = => 𝑠𝑖𝑚𝑝𝑙𝑖𝑓𝑖𝑒𝑑 𝑓𝑟𝑎𝑐𝑡𝑖𝑜𝑛

7 8 56 56 56

**Remember! You can not add the fractions until their denominators are same.**

1. **fraction operator \* (const fraction & f);** overload multiplication operator (\*). This function should store the product of calling object and the parameter **f** in a newly created object and return the simplest form of object to the calling function. e.g.,

 𝑠𝑖𝑚𝑝𝑙𝑖𝑓𝑖𝑒𝑑 𝑓𝑟𝑎𝑐𝑡𝑖𝑜𝑛 

1. bool isProper() const;

This function should return true if the fraction is proper, false otherwise. A proper fraction is one in which numerator is greater than denominator.

1. static int getCount();

This function should return the count of total number of fractions.

**Initialize the static data member by 0 outside the class.**

1. bool operator > (const fraction & f); overload relational operator (>). This function should return true if the calling fraction is greater than the parameter **f.** if the denominators of both the fractions are same then simply check the numerator.

1. void show() const

This function should display the data of calling object.

**Write a main function to test the functionalities of all the member functions of fraction ADT.**