**Set VS Multiset VS Memset**

#include <set>

*Set sort element from lower to upper*

set <int, greater <int> > s1;

this sort element to upper to lower

set<int> s1 ;

***// insert elements in random order***

s1.insert(40);

s1.insert(30);

***// you cannot insert same number***

s1.insert(40);

this will not be taken because it’s already taken

***// printing set s1***

set<int>::iterator p ;

for(p=s1.begin() ; p!= s1.end() ; p++)

{

cout<<\*p<<" ";

}

***// assigning the elements from s1 to s2***

set <int> s2(s1.begin(), s1.end());

***// Random input taking***

cin>>p;

s.insert(p); //Inserts an integer x into the set s

s.erase(p); //Erases an integer val from the set s.

***// remove all elements up to 30 in s2***

s2.erase(s2.begin(), s2.find(30));

***//finding element***

cin>>p;

it=s.find(p);

if(it!=s.end()) cout<<"Yes";

else cout<<"No";

**NOTE :**

set<int>::iterator itr=s.find(val); //Gives the iterator to the element val if it is found otherwise returns s.end() .

**Ex**: set<int>::iterator itr=s.find(100); //If 100 is not present then it==s.end().

**Multiset :**

The essential difference between the set and the multiset is that in a set the keys must be unique, while a multiset permits duplicate keys. ... In both sets and multisets, the sort order of components is the sort order of the keys, so the components in a multiset that have duplicate keys may appear in any order.

multiset <int > s1 ;

**Memset :**

#include<cstring> // for using Memset

bool prime[n+1];

memset(prime , true , sizeof(prime));

char str[] = "Hello World";

memset(str, 'o', 6); //take n = 6

int arr[6];

memset(arr , 0 , 6);

char str[] = "almost every programmer should know memset!";

memset (str,'-',6)

Output : ------ every programmer should know memset!