```
#include<iostream>
#include<conio.h>
#include<vector>
#include<queue>
#include<deque>
#include<array>
#include<list>
#include<string>
#include<numeric>
#include<algorithm>
#include<math.h>
using namespace std;
/*
        - adjacent_find()   ex-1
        - all_of()  ex-2
        - any_of() ex-3
        - sort()  //sorting.cpp
        - sort() in reverse order //sorting.cpp
        - binary_search() ex-4
        - lower_bound() ex-5
        - upper_bound() ex-5
        - max_element() ex-6
        - min_element() ex-6
        - for_each() ex-7
        - generate() ex-8
  - generate_n() ex-9
  - count() ex-10
        - count_if() ex-10
  - find() ex-11
        - find_if() ex-12
        - find_first_of() ex-13
        - equal() ex-14
        - equal_range() ex-15
        - fill() ex-16
  - merge() ex-17
  - remove()  ex-18
        - remove_if() ex-19
  - reverse() ex-20
  - replace() ex-21
        - replace_if() ex-22
        - rotate() ex-23
        - search() ex-24
  - unique() ex-25
  - is_sorted() ex-26
        - is_sorted_until() ex-27


*/
void example1() //adjacent_find()
{
  /*
```

```
  Searches the range [first,last) for the first occurrence of two consecutive elements that match,
  and returns an iterator to the first of these two elements, or last if no such pair is found.
  */
  vector<int>v1={10,4,4,13,7,7,21,15,11,11,11,20};
  int x=*adjacent_find(v1.begin(),v1.end());
  cout<<"Pair found with element value="<<x;
  vector<int>v2={10,4,13,7,21,15,11,20};
  vector<int>::iterator it=adjacent_find(v2.begin(),v2.end());
  if(it==v2.end())
   cout<<"\nNo such pair found";
}
void example2() //all_of()
{
  /*
     This function operates on whole range of array elements and can save time to run a loop
     to check each elements one by one. It checks for a given property on every element and
     returns true when each element in range satisfies specified property, else returns false.
  */
  vector<int> v1= { 10,20,14,50,18,6,12};
  if(all_of(v1.begin(),v1.end(),[](int a) -> int { return a%2==0;}))
    cout<<"All numbers are even";
  else
    cout<<"Not all numbers are even";

}
void example3() //any_of()
{
  /*
  Returns true if pred returns true for any of the elements in the range [first,last), and false
otherwise
  */
  vector<int> v1= { 10,20,14,5,18,6,12};
  if(any_of(v1.begin(),v1.end(),[](int a) -> int { return a%2==1;}))
    cout<<"At least one number is odd";
  else
    cout<<"All numbers are even";
}
void example4() //binary_search()
{
  /*
  This searching only works when container is sorted.
  This function returns boolean true if the element is present in the container, else returns false.
  */
  vector<int>v1={1,4,4,6,7,7,21,53,112,117,119,200};
  if(binary_search(v1.begin(),v1.end(),10))
    cout<<"Element found";
  else
    cout<<"Element not found";
}
void example5() //lower_bound() and upper_bound()
{
```

```
  /*
     Returns pointer to "position of num" if container contains first occurrence of num.
     Returns pointer to "first position of num" if container contains multiple occurrence of num.
     Returns pointer to "position of next higher number than num" if container does not contain
      occurrence of num.
  */
  vector<int>v1={1,4,4,6,7,7,21,53,112,117,119,200};
  vector<int>::iterator it=lower_bound(v1.begin(),v1.end(),9);
  cout<<"Element just greater than or equal to 9 is at index: "<<it-v1.begin();
  cout<<"\nand its value is "<<*it;
  /*
     Returns pointer to "position of next higher number than num" if container contains first
      occurrence of num.
     Returns pointer to "first position of next higher number than last occurrence of num"
      if container contains multiple occurrence of num.
     Returns pointer to "position of next higher number than num" if container does not contain
      occurrence of num.
  */
  vector<int>v2={1,4,4,6,7,7,21,53,112,117,119,200};
  it=upper_bound(v2.begin(),v2.end(),7);
  cout<<"Element just greater than 7 is at index: "<<it-v2.begin();
  cout<<"\nand its value is "<<*it;

}
void example6() //max_element() and min_element()
{
  /*
   max_element() returns an iterator pointing to the element with the largest value in the range
[first, last).
   min_element()  returns an iterator pointing to the element with the smallest value in the range
[first, last).
   */
  vector<int>v1={21,4,4,6,7,7,121,53,110,11,19,20};
  cout<<"Max Element is "<<*max_element(v1.begin(),v1.end());
  cout<<"\nMin Element is "<<*min_element(v1.begin(),v1.end());
}
void example7() // for_each()
{
  /*
  for_each is a function based looping technique.
  This loop accepts a function which executes over each of the container elements
  */
  vector<int>v1={21,4,4,6,7,7,121,53,110,11,19,20};
  for_each(v1.begin(),v1.end(),[](int x) -> void { cout<<x-1<<" ";});
}
void example8() //generate()
{
  /*
     It is used to generate numbers based upon a generator function, and then,
     it assigns those values to the elements in the container in the range [first, last).
     The generator function has to be defined by the user, and
```

```
    it is called successively for assigning the numbers
  */
  vector<int>v1(10);
  generate(v1.begin(),v1.end(),[]() -> int {static int i;++i; return i*i;});
  for(int num:v1)
  {
    cout<<num<<" ";
  }
}
void example9() //generate_n()
{
  /*
    there can be a scenario, where we want to assign values only to
    the first n elements, for that we have another STL algorithm std::generate_n
  */
  vector<int>v1(10);
  generate_n(v1.begin(),5,[]() -> int {static int i;++i; return i*i;});
  generate_n(v1.begin()+5,5,[]() -> int {static int j;++j; return j*j*j;});
  for(int num:v1)
  {
    cout<<num<<" ";
  }
}
void example10() //count() and count_if()
{
  /*
   It returns number of occurrences of an element in a given range
  */
  vector<int>v1={21,4,4,6,7,7,121,53,110,11,19,20};
  cout<<count(v1.begin(),v1.end(),4)<<endl;
  /*
  It is used to get the number of elements in a specified range which satisfy a condition.
  */
  cout<<count_if(v1.begin(),v1.end(),[](int x) -> bool {return x>15;});
}
void example11() //find()
{
 /*
  Finds the element in the given range of numbers.
  Returns an iterator to the first element in the range [first,last) that compares equal to val.
  If no such element is found, the function returns last.
 */
 vector<int>v1={21,4,4,6,7,7,121,53,110,11,19,20};
 vector<int>::iterator it=find(v1.begin(),v1.end(),110);
 if(it==v1.end())
   cout<<"Element not found";
 else
  cout<<"Element found at index "<<it-v1.begin();
}
void example12() //find_if()
{
```

```
 /*
   Returns an iterator to the first element in the range [first, last] for which
   pred(Unary Function) returns true
 */
 vector<int>v1={21,5,5,6,7,7,121,53,110,11,19,20};
 vector<int>::iterator it=find_if(v1.begin(),v1.end(),[](int x) ->bool {int k=(int)sqrt(x); cout<<k<<endl;
return k*k==x;});
 if(it==v1.end())
   cout<<"Element not found";
 else
   cout<<"Element "<<*it<<" found at index "<<it-v1.begin();
}
void example13() //find_first_of()
{
   /*
   it is used to compare elements between two containers.
   It compares all the elements in a range [first1,last1) with the elements in the range [first2,last2),
    and if any of the elements present in the second range is found in the first one ,
    then it returns an iterator to that element.
   If there are more than one element common in both the ranges,
    then an iterator to the first common element present in the first container is returned.
   In case there is no match, then iterator pointing to last1 is returned.
   */
   vector<int>v1={21,5,5,6,7,7,121,53,110,11,19,20};
   vector<int>v2={80,71,50,35,18,63,121,5};
   vector<int>::iterator it;
   it=find_first_of(v1.begin(),v1.end(),v2.begin(),v2.end());
   cout<<*it;
}
void example14() //equal()
{
   /*
   helps to compares the elements within the range [first_1,last_1) with those
    within range beginning at first_2.
   It returns boolean value
   */
   vector<int>v1={1,2,3,4,5,6};
   vector<int>v2={2,3,4};
   if(equal(v2.begin(),v2.end(),v1.begin()+1))
     cout<<"Equal";
   else
     cout<<"Not Equal";
}
void example15() //equal_range()
{
   /*
   It is used to find the sub-range within a given range [first, last) that has all
    the elements equivalent to a given value.
   It returns the initial and the final bound of such a sub-range.
   This function requires the range to be either sorted or partitioned according
    to some condition such that all the elements for which the condition evaluates
```

```
    to true are to the left of the given value and rest all are to its right
   */
   vector<int>v1={10,10,20,30,30,40,40,40,50,60,60,70,70,70,80};
   pair <vector<int>::iterator,vector<int>::iterator> p;
   p=equal_range(v1.begin(),v1.end(),40);
   cout << "40 is present in the sorted vector from index "<< (p.first - v1.begin()) << " till "<<
(p.second - v1.begin());
}
void example16() //fill()
{
   /*
   The 'fill' function assigns the value 'val' to all the elements in the range [begin, end),
    where 'begin' is the initial position and 'end' is the last position.
   Notice carefully that 'begin' is included in the range but 'end' is NOT included.
   */
   vector<int>v1={10,10,20,30,30,40,40,40,50,60,60,70,70,70,80};
   fill(v1.begin()+1,v1.begin()+5,100);
   for(int num:v1)
      cout<<num<<" ";

}
void example17() //merge()
{
 /*
  Combines the elements in the sorted ranges [first1,last1) and [first2,last2),
   into a new range beginning at result with all its elements sorted.
 */
 vector<int>v1={10,20,30,40};
 vector<int>v2={2,5,11,18,25};
 vector<int>v3(9);
 merge(v1.begin(),v1.end(),v2.begin(),v2.end(),v3.begin());
 for(int num:v3)
      cout<<num<<" ";
}
void example18() //remove()
{
   /*
   It removes value from range.
   Transforms the range [first,last) into a range with all the elements that compare equal to val
removed,
    and returns an iterator to the new end of that range.
   */
   vector<int>v1={11,2,5,11,18,25};
   vector<int>::iterator it,newEnd;
   newEnd=remove(v1.begin(),v1.end(),11);
   for(int num:v1)
      cout<<num<<" ";
   cout<<endl;
   for(it=v1.begin();it!=newEnd;it++)
      cout<<*it<<" ";
}
```

```
void example19() //remove_if()
{
   /*
    remove_if() function is used to eliminate all the elements that satisfy a predicate from a given
     range [first, last) without disturbing the order of the remaining elements.
   */
   vector<int>v1={10,3,4,4,4,5,5,77,8,5,5,2};
   vector<int>::iterator it,newEnd;
   newEnd=remove_if(v1.begin(),v1.end(),[](int x) -> bool{return x%2==0;});
   for(int num:v1)
      cout<<num<<" ";
   cout<<endl;
   for(it=v1.begin();it!=newEnd;it++)
      cout<<*it<<" ";
}
void example20() //reverse()
{
   /*
   It reverses the order of the elements in the range [first, last) of any container.
   */
   vector<int>v1={10,3,4,4,4,5,5,77};
   reverse(v1.begin(),v1.end());
   for(int num:v1)
      cout<<num<<" ";
   cout<<endl;
}
void example21() //replace()
{
   /*
   Assigns new_value to all the elements in the range [first, last) that compare to old_value.
   */
   vector<int>v1={10,3,4,4,4,5,5,77};
   replace(v1.begin(),v1.end(),4,63);
   for(int num:v1)
      cout<<num<<" ";
   cout<<endl;
}
void example22() //replace_if()
{
   /*
   Assigns new_value to all the elements in range [first, last) for which pred returns true.
   */
   vector<int>v1={10,3,4,4,4,5,5,77};
   replace_if(v1.begin(),v1.end(),[](int x) ->bool{return x>5;},0);
   for(int num:v1)
      cout<<num<<" ";
   cout<<endl;
}
void example23() //rotate()
{
   /*
```

```
   It rotates the order of the elements in the range [first,last], in such a way that the element pointed
    by middle becomes the new first element.
   */
   vector<int>v1={10,3,4,4,4,5,5,77};
   rotate(v1.begin(),v1.begin()+2,v1.end());
   for(int num:v1)
      cout<<num<<" ";
   cout<<endl;
}
void example24() //search()
{
   /*
   It searches the sequence [first1, last1) for the first occurrence of the subsequence defined by
[first2, last2),
    and returns an iterator to its first element of the occurrence, or last1 if no occurrences are found.
   */
   vector<int>v1={11,44,22,77,33,99,66,55,88};
   vector<int>v2={99,66,55};

   vector<int>::iterator it;
   it=search(v1.begin(),v1.end(),v2.begin(),v2.end());
   if(it!=v1.end())
      cout<<"Subsequence found at index "<<it-v1.begin();
   else
      cout<<"Subsequence not found";
}
void example25() //unique()
{
   /*
   It does not delete all the duplicate elements, but it removes duplicacy by just replacing those
elements
    by the next element present in the sequence which is not duplicate to the current element being
   replaced. All the elements which are replaced are left in an unspecified state.

   Another interesting feature of this function is that it does not changes the size of the container
after
    deleting the elements, it just returns a pointer pointing to the new end of the container
   */
   vector<int>v1={11,44,22,77,33,33,33,55,88};
   vector<int>::iterator it,newEnd;
   newEnd=unique(v1.begin(),v1.end());
   for(int num:v1)
      cout<<num<<" ";
   cout<<endl;
   for(it=v1.begin();it!=newEnd;it++)
      cout<<*it<<" ";
}
void example26() //is_sorted()
{
   /*
   It checks if the elements in range [first, last] are sorted in ascending order.
```

```
   */
   vector<int>v1={11,34,22,25,33,33,33,55,88};
   if(is_sorted(v1.begin(),v1.end()))
      cout<<"Yes vector is sorted";
   else
      cout<<"No, vector is not sorted";

}
void example27() //is_sorted_until()
{
   /*
    It is used to find out the first unsorted element in the range [first, last).
    It returns an iterator to the first unsorted element in the range, so all the elements in between
first
     and the iterator returned are sorted.
    */
    vector<int>v1={11,14,22,77,33,33,33,55,88};
    vector<int>::iterator it;
    it=is_sorted_until(v1.begin(),v1.end());
    cout<<*it<<endl;
    cout<<"Number of sorted elements until the first unsorted one is "<<it-v1.begin();

}
int main()
{
   //example1(); //adjacent_find()
   //example2(); //all_of()
   //example3(); //any_of()
   //example4(); //binary_search()
   //example5(); //lower_bound() and upper_bound()
   //example6(); //max_element() and min_element()
   //example7(); //for_each()
   //example8(); //generate()
   //example9(); //generate_n()
   //example10(); //count()  and count_if()
   //example11(); //find()
   //example12(); //find_if()
   //example13(); //find_first_of()
   //example14(); //equal()
   //example15(); //equal_range()
   //example16(); //fill()
   //example17(); //merge()
   //example18(); //remove()
   //example19(); //remove_if()
   //example20();   //reverse()
   //example21();  //replace()
         //example22(); //replace_if()
         //example23(); //rotate()
         //example24();  //search()
   //example25(); //unique()
   //example26(); //is_sorted()
```

```
        example27(); //is_sorted_until()
    getch();
}
```