Vector

```
// constructing vectors
#include <iostream>
#include <vector>
using namespace std;

int main () {
    // constructors used in the same order as described above:
    vector<int> first;                                // empty vector of ints
    vector<int> second (4,100);                       // four ints with value 100
    vector<int> third (second.begin(),second.end());  // iterating through second
    vector<int> fourth (third);                       // a copy of third

    // the iterator constructor can also be used to construct from arrays:
    int myints[] = {16,2,77,29};
    vector<int> fifth (myints, myints + sizeof(myints) / sizeof(int) );

    cout << "The contents of fifth are:";
    for (unsigned i=0; i < fifth.size(); i++) cout << " " << fifth[i]; cout << endl;

    return 0;
}
```

////////3d array of vector

```
#include <iostream>
#include <vector>
using namespace std;

int main () {
    // use of [] and ()
    vector<int> v1[100]; // creates an array of vectors, i.e. 100 vectors
    vector<int> v2(100); // creates 1 vector of size 100

    // creating a 2D array 100x2 where each element is a vector,
    // so in total, it is a 3D structure, as vector itself is 1D
    vector<int> v3[100][2];
    return 0;
}
```

/////2d vector output

```
#include <iostream>
#include <vector>
using namespace std;

int main () {
    vector< vector< int > > V2D;

    // creating a 2D triangular structure
    for(int i = 0; i < 10; i++) {
        vector< int > temp;
        for(int j = 0; j <= i; j++) {
            temp.push_back(i);
        }
        V2D.push_back(temp);
    }

    // using iterator
    cout << "using iterator:\n";
    vector< vector< int > > :: iterator outer;
    vector< int > :: iterator inner;
    for(outer = V2D.begin(); outer != V2D.end(); outer++) {
        for(inner = outer->begin(); inner != outer->end(); inner++) {
            cout << *inner << ' ';
        }
        cout << '\n';
    }

    // using index
    cout << "\nusing indexes:\n";
    for(unsigned i = 0; i < V2D.size(); i++) {
        for(unsigned j = 0; j < V2D[i].size(); j++) {
            cout << V2D[i][j] << ' ';
        }
        cout << '\n';
    }
    return 0;
}
```

/////vecotor function

পুশব্যাক, পপব্যাক, সাইজ, এম্পটি

আগের কোডটায় আমরা দেখলাম পুশব্যাক দিয়ে ভেক্টরে ভ্যালু ইন্সার্ট করা হচ্ছে, তাই আর নতুন করে সেটা দেখনোর কিছু নাই, push_back() ফাংশনের সাহায্যে ভেক্টরের শেষে একি ধরনের একটা আইটেম অ্যাড করা যায়, আর pop_back() দিয়ে ভেক্টরের শেষ থেকে একটা আইটেম হাওয়া করে দেওয়া যায়। অন্যান্য STL মেম্বারের মত ভেক্টরের size() আর empty() ফাংশন দুইটাও আইডেন্টিক্যাল। নিচের কোডে এগুলোর ব্যবহার দেখানো হলঃ

```
#include <iostream>
#include <vector>
using namespace std;

int main () {
    vector< int > simple;

    for(int i = 1; i < 10; i++) {
        simple.push_back(i*100);
        cout << "inserting: " << simple.back() << endl;
    }
    cout << "-------------------\n";

    while(!simple.empty()) {
        cout << "size: " << simple.size();
        cout << " last element: " << simple.back() << endl;
        simple.pop_back();
    }
    cout << "vector empty\n";
    return 0;
}
```

//////রিসাইজ, ইরেজ, ক্লিয়ার এবং রিসাইজ নিয়ে কিছু কাহিনীঃ

ভেক্টরের সাইজ মডিফায় করা যায় এরকম কিছু মেম্বার হল resize(), erase(), clear(), এদের মধ্যে clear() এর ব্যবহার অন্য যে কোন কন্টেইনার ক্লাসের মতই, সব এলিমেনত ডিলিট করে দেয়, ফলাফম সাইজ ০ হয়ে যায়। আর resize() ফাংশন দিয়ে ভেক্টরের সাইজ পরিবর্তন করা যায়। erase() এর কাজ কোন এলিমেন্ট বা একটা রেঞ্জ ডিলিট করা যায়। erase() এর দুইটা ফরম্যাট আছে, erase(iterator pos), erase(iterator first, iterator last), অর্থাৎ কোন ভ্যালুর সাপেক্ষে ডিলিট করা যায় না, তার ইটারেটর পজিশন দিতে হবে, আর দ্বিতীয় ধরনে ভেক্টরের [first, last) এই রেঞ্জের সব আইটেম ডিলিট করে দিবে। বলাই বাহুল্য, আজগুবি ইটারেটর দিলে রানটাইম এররর খেয়ে বসে থাকতে হবে...

```
#include <iostream>
#include <vector>
using namespace std;

int main () {
    unsigned int i;
    vector<unsigned int> myvector;
    // set some values (from 1 to 10)
    for (i=1; i<=10; i++) myvector.push_back(i);
    // erase the 6th element
    myvector.erase (myvector.begin()+5);
    // erase the first 3 elements:
    myvector.erase (myvector.begin(),myvector.begin()+3);
    cout << "myvector contains:";
    for (i=0; i<myvector.size(); i++)
```

```
        cout << " " << myvector[i];
    cout << endl;

    // clear all
    myvector.clear();
    cout << "size: " << myvector.size() << endl;

    // resize and then check size
    myvector.resize(10);
    cout << "size: " << myvector.size() << endl;
    return 0;
}
```
কিছু বিষয়ে এখানে সতর্ক হতে হবে, যেমনঃ resize() ফাংশনটা ভেক্টরের আগের অবস্থাত কোন তোয়াক্কা না করেই তার সাইজ চেঞ্জ করবে, ফলে কিছু এলিমেন্ট বাদও পড়তে পারে, আবার কিছু জায়গা খালিও থাকতে পারে। কিন্তু ভেক্টরের সাইজ চেক করলে নতুন সাইজ এ পাওয়া যাবে, যদিও, সেটা হয়তো একটা খালি ভেক্টর ছিল। তাই, resize() এর পরে push_back() ব্যবহার করলে কোডার যাই আশা করুক না কেন, সে রিসাজকৃত স্পেসের পরেই পুশ করবে। যেমন, মনে করি, ভেক্টরে ছিল ১৮ টা আইটেম, আমি সেটাকে রিসাইজ করলাম ৫০ এ। তখন push_back() করলে সে ১৯ তম পজিশনে করবে না, করবে ৫১ তম পজিশনে (মানে ইন্ডেক্স ৫০)। একই ভাবে যদি ১২ রে রিসাইজ করতাম, তাহলেও সে ১৯ এ না করে ১৩ তে পুশ করতো। সাধারনত clear() এর অল্টারনেট হিসাবে resize() ব্যবহার করা যায়, তবে এর সাথে push_back() ব্যবহার না করাই ভাল। নিচের কোডঃ

//////////////pair

```
/ CPP Program to demonstrate make_pair()
// function in pair
#include <iostream>
#include <utility>
using namespace std;

// Driver Code
int main()
{
    pair<int, char> PAIR1;
    pair<string, double> PAIR2("GeeksForGeeks", 1.23);
    pair<string, double> PAIR3;

    PAIR1.first = 100;
    PAIR1.second = 'G';

    PAIR3 = make_pair("GeeksForGeeks is Best", 4.56);

    cout << PAIR1.first << " ";
    cout << PAIR1.second << endl;

    cout << PAIR2.first << " ";
    cout << PAIR2.second << endl;

    cout << PAIR3.first << " ";
    cout << PAIR3.second << endl;

    return 0;
}

/////////pair sort

Bool cmp(pair<int,int> &a, pair<int,int> &b){
if (a.first == b.first)
    Return a.second< b.second;
Return a.first<a.second;
}
```

////vector of pair

```
// C++ program to demonstrate vector of pairs
#include<bits/stdc++.h>
using namespace std;

int main()
{
    //declaring vector of pairs
    vector< pair <int,int> > vect;

    // initialising 1st and 2nd element of
    // pairs with array values
    int arr[] = {10, 20, 5, 40 };
    int arr1[] = {30, 60, 20, 50};
    int n = sizeof(arr)/sizeof(arr[0]);

    // Entering values in vector of pairs
    for (int i=0; i<n; i++)
        vect.push_back( make_pair(arr[i],arr1[i]) );

    // Printing the vector
    for (int i=0; i<n; i++)
    {
        // "first" and "second" are used to access
        // 1st and 2nd element of pair respectively
        cout << vect[i].first << " "
```

```
              << vect[i].second << endl;
    }

    return 0;
}
```

///////set

```
// C++ program to demonstrate various functions of
// STL
#include <iostream>
#include <iterator>
#include <set>
using namespace std;

int main()
{
    // empty set container
    set<int, greater<int> > s1;

    // insert elements in random order
    s1.insert(40);
    s1.insert(30);
    s1.insert(60);
    s1.insert(20);
    s1.insert(50);

    // only one 50 will be added to the set
    s1.insert(50);
    s1.insert(10);

    // printing set s1
    set<int, greater<int> >::iterator itr;
    cout << "\nThe set s1 is : \n";
    for (itr = s1.begin(); itr != s1.end(); itr++) {
        cout << *itr << " ";
    }
    cout << endl;

    // assigning the elements from s1 to s2
    set<int> s2(s1.begin(), s1.end());
    // print all elements of the set s2
    cout << "\nThe set s2 after assign from s1 is : \n";
    for (itr = s2.begin(); itr != s2.end(); itr++) {
        cout << *itr << " ";
    }
    cout << endl;

    // remove all elements up to 30 in s2
    cout << "\ns2 after removal of elements less than 30 "
            ":\n";
    s2.erase(s2.begin(), s2.find(30));
    for (itr = s2.begin(); itr != s2.end(); itr++) {
        cout << *itr << " ";
    }

    // remove element with value 50 in s2
    int num;
    num = s2.erase(50);
    cout << "\ns2.erase(50) : ";
    cout << num << " removed\n";
    for (itr = s2.begin(); itr != s2.end(); itr++) {
        cout << *itr << " ";
    }

    cout << endl;

    // lower bound and upper bound for set s1
    cout << "s1.lower_bound(40) : "
         << *s1.lower_bound(40) << endl;
    cout << "s1.upper_bound(40) : "
         << *s1.upper_bound(40) << endl;

    // lower bound and upper bound for set s2
    cout << "s2.lower_bound(40) : "
         << *s2.lower_bound(40) << endl;
    cout << "s2.upper_bound(40) : "
         << *s2.upper_bound(40) << endl;

    return 0;
}
```
Output:

```
The set s1 is :
```

```
60 50 40 30 20 10

The set s2 after assign from s1 is :
10 20 30 40 50 60

s2 after removal of elements less than 30 :
30 40 50 60
s2.erase(50) : 1 removed
30 40 60
s1.lower_bound(40) : 40
s1.upper_bound(40) : 30
s2.lower_bound(40) : 40
s2.upper_bound(40) : 60
```

//////stack

```cpp
// stack::push/pop/top/size/empty
#include <iostream>
#include <stack>
using namespace std;

int main ()
{
    stack< int > mystack;
    // lets push and pop some values
    for (int i=0; i<5; i++) mystack.push(i);

    cout << "Popping out elements...";
    while (!mystack.empty())
    {
        cout << " " << mystack.top();
        mystack.pop();
    }
    cout << endl;

    // changing the value of top
    mystack.push(100);
    mystack.top() += 1000;
    cout << "Now top is: " << mystack.top() << endl;
    mystack.pop();

    // not a good way to check if stack is empty
    if(mystack.size() == 0) cout << "Stack is empty" << endl;
    // better we do this
    if(mystack.empty()) cout << "Stack is empty" << endl;

    return 0;
}
```

/////queue

```
// queue::size/empty
#include <iostream>
#include <queue>
using namespace std;

int main ()
{
    queue<int> Q;

    // just push some elements
    for(int i = 0; i < 5; i++) Q.push(i*i);

    cout << "Queue has " << Q.size() << " elements" << endl;

    Q.pop(); // not a good way, first check for Q.empty()

    if(!Q.empty()) Q.pop(); // this is the porper way

    while(!Q.empty()) Q.pop(); // pop all element

    if(Q.size()==0) cout << "Q is empty" << endl; // not a good way
    if(Q.empty()) cout << "Q is empty" << endl; // this is the proper way

    return 0;
```

/////front(), back()

```
// queue::front/back
#include <iostream>
```

```
#include <queue>
using namespace std;

int main ()
{
    queue<int> myqueue;

    myqueue.push(77);
    myqueue.push(16);
    cout << "myqueue.front() = " << myqueue.front() << endl;
    cout << "myqueue.back() = " << myqueue.back() << endl;

    // modify front element
    myqueue.front() -= myqueue.back();
    cout << "myqueue.front() is now " << myqueue.front() << endl;

    // modify back element
    myqueue.back() += myqueue.front();
    cout << "myqueue.back() is now " << myqueue.back() << endl;

    return 0;
}
```

////map

https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/

////////deque

```
// CPP Program to implement Deque in STL
#include <deque>
#include <iostream>

using namespace std;

void showdq(deque<int> g)
{
    deque<int>::iterator it;
    for (it = g.begin(); it != g.end(); ++it)
        cout << '\t' << *it;
    cout << '\n';
}
```

```
int main()
{
    deque<int> gquiz;
    gquiz.push_back(10);
    gquiz.push_front(20);
    gquiz.push_back(30);
    gquiz.push_front(15);
    cout << "The deque gquiz is : ";
    showdq(gquiz);

    cout << "\ngquiz.size() : " << gquiz.size();
    cout << "\ngquiz.max_size() : " << gquiz.max_size();

    cout << "\ngquiz.at(2) : " << gquiz.at(2);
    cout << "\ngquiz.front() : " << gquiz.front();
    cout << "\ngquiz.back() : " << gquiz.back();

    cout << "\ngquiz.pop_front() : ";
    gquiz.pop_front();
    showdq(gquiz);

    cout << "\ngquiz.pop_back() : ";
    gquiz.pop_back();
    showdq(gquiz);

    return 0;
}
```

Output:

```
The deque gquiz is :     15    20    10    30


gquiz.size() : 4

gquiz.max_size() : 4611686018427387903

gquiz.at(2) : 10

gquiz.front() : 15

gquiz.back() : 30

gquiz.pop_front() :     20    10    30


gquiz.pop_back() :     20    10
```

.//////priority queue

```
// C++ program to demonstrate the use of priority_queue
#include <iostream>
#include <queue>
using namespace std;

// driver code
int main()
{
    int arr[6] = { 10, 2, 4, 8, 6, 9 };

    // defining priority queue
    priority_queue<int> pq;

    // printing array
    cout << "Array: ";
    for (auto i : arr) {
        cout << i << ' ';
    }
    cout << endl;
    // pushing array sequentially one by one
    for (int i = 0; i < 6; i++) {
        pq.push(arr[i]);
    }

    // printing priority queue
    cout << "Priority Queue: ";
    while (!pq.empty()) {
        cout << pq.top() << ' ';
        pq.pop();
    }

    return 0;
}
```

Output

```
Array: 10 2 4 8 6 9

Priority Queue: 10 9 8 6 4 2
```

///////another pririty queue

```
// C++ program to demonstrate
// min heap for priority queue
#include <iostream>
#include <queue>
using namespace std;

void showpq(
    priority_queue<int, vector<int>, greater<int> > g)
{
    while (!g.empty()) {
        cout << ' ' << g.top();
        g.pop();
    }
    cout << '\n';
}

void showArray(int* arr, int n)
{
    for (int i = 0; i < n; i++) {
        cout << arr[i] << ' ';
    }
    cout << endl;
}

// Driver Code
int main()
{
    int arr[6] = { 10, 2, 4, 8, 6, 9 };
    priority_queue<int, vector<int>, greater<int> > gquiz(
        arr, arr + 6);

    cout << "Array: ";
      showArray(arr, 6);

    cout << "Priority Queue : ";
    showpq(gquiz);

    return 0;
}
```

Output

```
Array: 10 2 4 8 6 9
```

```
Priority Queue :  2 4 6 8 9 10
```

///////list

```
// C++ program to demonstrate the implementation of List
#include <iostream>
#include <iterator>
#include <list>
using namespace std;

// function for printing the elements in a list
void showlist(list<int> g)
{
    list<int>::iterator it;
    for (it = g.begin(); it != g.end(); ++it)
        cout << '\t' << *it;
    cout << '\n';
}

// Driver Code
int main()
{

    list<int> gqlist1, gqlist2;

    for (int i = 0; i < 10; ++i) {
        gqlist1.push_back(i * 2);
        gqlist2.push_front(i * 3);
    }
    cout << "\nList 1 (gqlist1) is : ";
    showlist(gqlist1);

    cout << "\nList 2 (gqlist2) is : ";
    showlist(gqlist2);

    cout << "\ngqlist1.front() : " << gqlist1.front();
    cout << "\ngqlist1.back() : " << gqlist1.back();

    cout << "\ngqlist1.pop_front() : ";
    gqlist1.pop_front();
    showlist(gqlist1);
    cout << "\ngqlist2.pop_back() : ";
    gqlist2.pop_back();
    showlist(gqlist2);

    cout << "\ngqlist1.reverse() : ";
    gqlist1.reverse();
    showlist(gqlist1);

    cout << "\ngqlist2.sort(): ";
    gqlist2.sort();
    showlist(gqlist2);

    return 0;
}
```

Output

```
List 1 (gqlist1) is :    0    2    4    6    8    10    12    14    16    18


List 2 (gqlist2) is :    27    24    21    18    15    12    9    6    3    0


gqlist1.front() : 0
gqlist1.back() : 18
gqlist1.pop_front() :    2    4    6    8    10    12    14    16    18


gqlist2.pop_back() :    27    24    21    18    15    12    9    6    3


gqlist1.reverse() :    18    16    14    12    10    8    6    4    2


gqlist2.sort():    3    6    9    12    15    18    21    24    27
```

/////map

```
#include <iostream>
```

```
#include <map>

int main()
{
  // Create a map of strings to integers
map<string, int> map;

  // Insert some values into the map
  map["one"] = 1;
  map["two"] = 2;
  map["three"] = 3;

  // Print the size of the map
  std::cout << "Size of map: " << map.size() << std::endl;

  return 0;
}
```

////////another

```
// CPP Program to demonstrate the implementation in Map
// divyansh mishra --> divyanshmishra101010
#include <iostream>
#include <iterator>
#include <map>
using namespace std;

int main()
{

    // empty map container
    map<int, int> gquiz1;

    // insert elements in random order
    gquiz1.insert(pair<int, int>(1, 40));
    gquiz1.insert(pair<int, int>(2, 30));
    gquiz1.insert(pair<int, int>(3, 60));
    gquiz1.insert(pair<int, int>(4, 20));
    gquiz1.insert(pair<int, int>(5, 50));
    gquiz1.insert(pair<int, int>(6, 50));

      gquiz1[7]=10;     // another way of inserting a value in a map
```

```cpp
    // printing map gquiz1
    map<int, int>::iterator itr;
    cout << "\nThe map gquiz1 is : \n";
    cout << "\tKEY\tELEMENT\n";
    for (itr = gquiz1.begin(); itr != gquiz1.end(); ++itr) {
        cout << '\t' << itr->first << '\t' << itr->second
             << '\n';
    }
    cout << endl;

    // assigning the elements from gquiz1 to gquiz2
    map<int, int> gquiz2(gquiz1.begin(), gquiz1.end());

    // print all elements of the map gquiz2
    cout << "\nThe map gquiz2 after"
         << " assign from gquiz1 is : \n";
    cout << "\tKEY\tELEMENT\n";
    for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr) {
        cout << '\t' << itr->first << '\t' << itr->second
             << '\n';
    }
    cout << endl;

    // remove all elements up to
    // element with key=3 in gquiz2
    cout << "\ngquiz2 after removal of"
            " elements less than key=3 : \n";
    cout << "\tKEY\tELEMENT\n";
    gquiz2.erase(gquiz2.begin(), gquiz2.find(3));
    for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr) {
        cout << '\t' << itr->first << '\t' << itr->second
             << '\n';
    }

    // remove all elements with key = 4
    int num;
    num = gquiz2.erase(4);
    cout << "\ngquiz2.erase(4) : ";
    cout << num << " removed \n";
    cout << "\tKEY\tELEMENT\n";
    for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr) {
        cout << '\t' << itr->first << '\t' << itr->second
             << '\n';
    }

    cout << endl;

    // lower bound and upper bound for map gquiz1 key = 5
    cout << "gquiz1.lower_bound(5) : "
         << "\tKEY = ";
    cout << gquiz1.lower_bound(5)->first << '\t';
    cout << "\tELEMENT = " << gquiz1.lower_bound(5)->second
         << endl;
    cout << "gquiz1.upper_bound(5) : "
         << "\tKEY = ";
    cout << gquiz1.upper_bound(5)->first << '\t';
    cout << "\tELEMENT = " << gquiz1.upper_bound(5)->second
         << endl;

    return 0;
}
```

Output

```
The map gquiz1 is :

    KEY    ELEMENT

    1     40

    2     30

    3     60

    4     20

    5     50

    6     50

    7     10



The map gquiz2 after assign from gquiz1 is :

    KEY    ELEMENT

    1     40

    2     30

    3     60
    4    20
    5    50
    6    50
    7    10


gquiz2 after removal of elements less than key=3 :
    KEY    ELEMENT
    3    60
    4    20
    5    50
    6    50
    7    10

gquiz2.erase(4) : 1 removed
    KEY    ELEMENT
    3    60
    5    50
    6    50
    7    10

gquiz1.lower_bound(5) :     KEY = 5        ELEMENT = 50
gquiz1.upper_bound(5) :     KEY = 6        ELEMENT = 50
```

//////string push back,pop back

```
// C++ Program to demonstrate the working of
// getline(), push_back() and pop_back()
#include <iostream>
```

```
#include <string> // for string class
using namespace std;

// Driver Code
int main()
{
    // Declaring string
    string str;

    // Taking string input using getline()
    getline(cin, str);

    // Displaying string
    cout << "The initial string is : ";
    cout << str << endl;

    // Inserting a character
    str.push_back('s');

    // Displaying string
    cout << "The string after push_back operation is : ";
    cout << str << endl;

    // Deleting a character
    str.pop_back();

    // Displaying string
    cout << "The string after pop_back operation is : ";
    cout << str << endl;

    return 0;
}
```

/////////some other function

```
// C++ Program to demonstrate the working of
// capacity(), resize() and shrink_to_fit()
#include <iostream>
#include <string> // for string class
using namespace std;

// Driver Code
int main()
{
    // Initializing string
    string str = "geeksforgeeks is for geeks";

    // Displaying string
    cout << "The initial string is : ";
    cout << str << endl;

    // Resizing string using resize()
    str.resize(13);

    // Displaying string
    cout << "The string after resize operation is : ";
    cout << str << endl;

    // Displaying capacity of string
    cout << "The capacity of string is : ";
    cout << str.capacity() << endl;

    // Displaying length of the string
    cout << "The length of the string is :" << str.length()
         << endl;



    // Displaying string
    cout << "The new capacity after shrinking is : ";
    cout << str.capacity() << endl;

    return 0;
}
////////////some other funcction


// C++ Program to demonstrate the working of
// copy() and swap()
#include <iostream>
#include <string> // for string class
using namespace std;

// Driver Code
int main()
{
    // Initializing 1st string
    string str1 = "geeksforgeeks is for geeks";

    // Declaring 2nd string
    string str2 = "geeksforgeeks rocks";


    // Displaying strings before swapping
    cout << "The 1st string before swapping is : ";
    cout << str1 << endl;
    cout << "The 2nd string before swapping is : ";
    cout << str2 << endl;

    // using swap() to swap string content
    str1.swap(str2);

    // Displaying strings after swapping
    cout << "The 1st string after swapping is : ";
    cout << str1 << endl;
    cout << "The 2nd string after swapping is : ";
    cout << str2 << endl;

    return 0;
}
```

An anagram of a string is another string that contains the same characters, only the order of characters can be different. For example, “abcd” and “dabc” are an anagram of each other.

Input: *str1 = “listen” str2 = “silent”*
Output: *“Anagram”*
Explanation: *All characters of “listen” and “silent” are the same.*
Input: *str1 = “gram” str2 = “arm”*
Output: *“Not Anagram”*

```
// C++ program to check whether two strings are anagrams
// of each other
#include <bits/stdc++.h>
using namespace std;

/* function to check whether two strings are anagram of
each other */
bool areAnagram(string str1, string str2)
{
    // Get lengths of both strings
    int n1 = str1.length();
    int n2 = str2.length();

    // If length of both strings is not same, then they
    // cannot be anagram
    if (n1 != n2)
        return false;

    // Sort both the strings
    sort(str1.begin(), str1.end());
    sort(str2.begin(), str2.end());

    // Compare sorted strings
    for (int i = 0; i < n1; i++)
        if (str1[i] != str2[i])
            return false;

    return true;
}

// Driver code
int main()
{
    string str1 = "gram";
    string str2 = "arm";

    // Function Call
    if (areAnagram(str1, str2))
        cout << "The two strings are anagram of each other";
    else
        cout << "The two strings are not anagram of each "
                "other";

    return 0;
}

//////sort an string

#include <bits/stdc++.h>
using namespace std;

// Function to sort the strings
// in lexicographical order
void Printlexiographically(int N, string arr[])
```

```
{
    // Sort the strings of the array
    sort(arr, arr + N);

    for (int i = 0; i < N; i++) {

        // Print each string of the array
        cout << arr[i] << '\n';
    }
}

// Driver Code
int main()
{
    string arr[] = { "batman", "bat", "apple" };
    int N = sizeof(arr) / sizeof(arr[0]);

    // Function Call
    Printlexiographically(N, arr);

    return 0;
}


////////string stream

#include<bits/stdc++.h>
using namespace std;

int main(){

string s, st;
getline(cin, s);
stringstream ss(s);

while(ss>>st){

cout<<st<<endl;
}

}

Input : I am a student
Output : i
Am
```

A
Student

Now Take an string “11 12 13 14”
Now save each string into an array

```
#include<bits/stdc++.h>
Using namespace std;

Int main()
{

string s;
Getline(cin, s)
Int ar[20];
Stringsteam ss(s);
Int p=0;
While(ss>>ar[p]){
P++;
}
For(int i=0;i<p;i++)cout<<ar[i]<<” “<<endl;
}
```