STL

The Standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators.

SORTING

Sorting

Data **sorting** is any process that involves arranging the data into some meaningful order.We generally sort the array in ascending or descending order.

Examples of sorting:-

A = {1, 6, 4, 3, 3, 2, 7}

A = {1, 2, 3, 3, 4, 6, 7} -- sorted in ascending

A = {7, 6, 4, 3, 3, 2, 1} -- sorted in descending

To learn more about sorting algorithms refer this link :-
https://www.geeksforgeeks.org/sorting-algorithms/#algo

Sorting in Competitive Programming

Time Complexity of mergesort and quicksort = O(n * log(n))

Sort STL :- a[n]

sort(a, a+n); -- ascending order

sort(a, a+n, greater<int>()); -- descending order

E.g. **a[8] = {8,9,7,6,4,5,10,1}**

Q. How to sort the complete array?

sort(a,a+8);

Ans : {1,4,5,6,7,8,9,10}

Q. How to sort the first four elements?

sort(a,a+4);

Ans : {6,7,8,9,4,5,10,1}

Q. How to sort the last four elements?

sort(a+4,a+8);

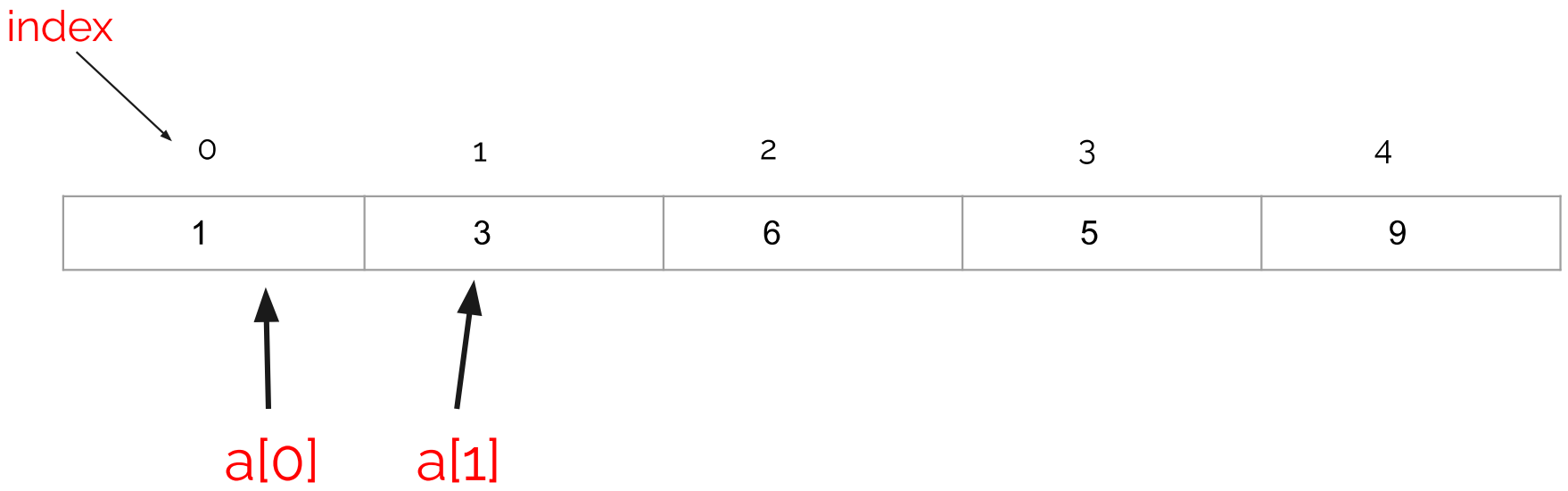
Ans : {8,9,7,6,1,4,5,10}

VECTOR

Vector

Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container.

```
int  n;
cin>>n;
int a[n];    n=5
```



```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   vector<int> v;
   int n;
   cin >> n;
   for(int i=0; i<n; i++)
   {
        v.push_back(i);
   }
   v.push_back(10);
     for(int i=0; i<v.size(); i++)
  {
    cout<<v[i]<<" ";
  }
cout<<endl;
v.pop_back( );
v.pop_back( );
int len=v.size( );
cout<<len<<endl;
for(int i=0; i<v.size(); i++)
{
    cout<<v[i]<<" ";
 }
  return 0;
}
```

INPUT

5

Output

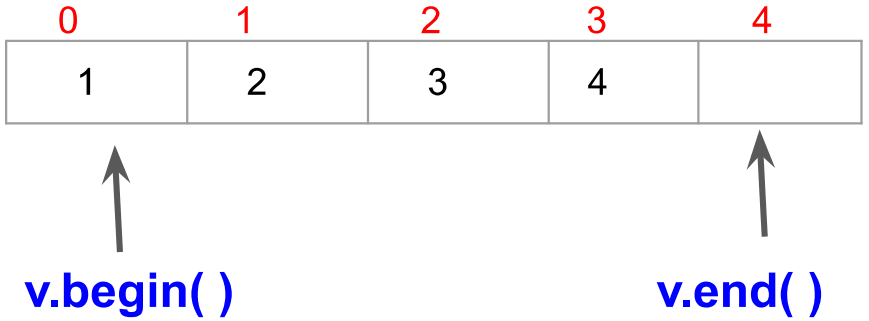
0 1 2 3 4 10

4

0 1 2 3

- push_back(x)
- pop_back()
- size( )
- begin( )
- end( )

vector <int> v;
v.size()=4;



sort(a, a+n);

sort(v.begin( ), v.end( ));

PAIR

Pair

Coordinates (x,y) = { (6,7) , (12,10) , (4,9) , (16, 8) , (13,17) }

x ->	6	12	4	16	13
y ->	7	10	9	8	17

If we sort the coordinates , x[] and y[]

x ->	4	6	12	13	16
y ->	7	8	9	10	17

Pair is used to combine two values of 2 same or different data types together. To access the elements, we use variable name with dot operator followed by first or second.

pair<data_type,data_type> p;

pair<string,int> p[n]; //(array of pair)

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long

int main() {
      pair<ll,ll> p[5];
      for(int i = 0 ; i < 5 ; i++) {
          ll a,b;
          cin >> a >> b;
          p[i].first = a;
          p[i].second = b;
      }
      sort(p,p+5);
```

```
for(int i = 0 ; i < 5 ; i++)
{
 cout << p[i].first << " "
<< p[i].second;
 cout << "\n";
 }
      return 0;
}
```

INPUT

```
1 2
6 5
4 6
3 8
5 10
```

OUTPUT

```
1 2
3 8
4 6
5 10
6 5
```

MAP

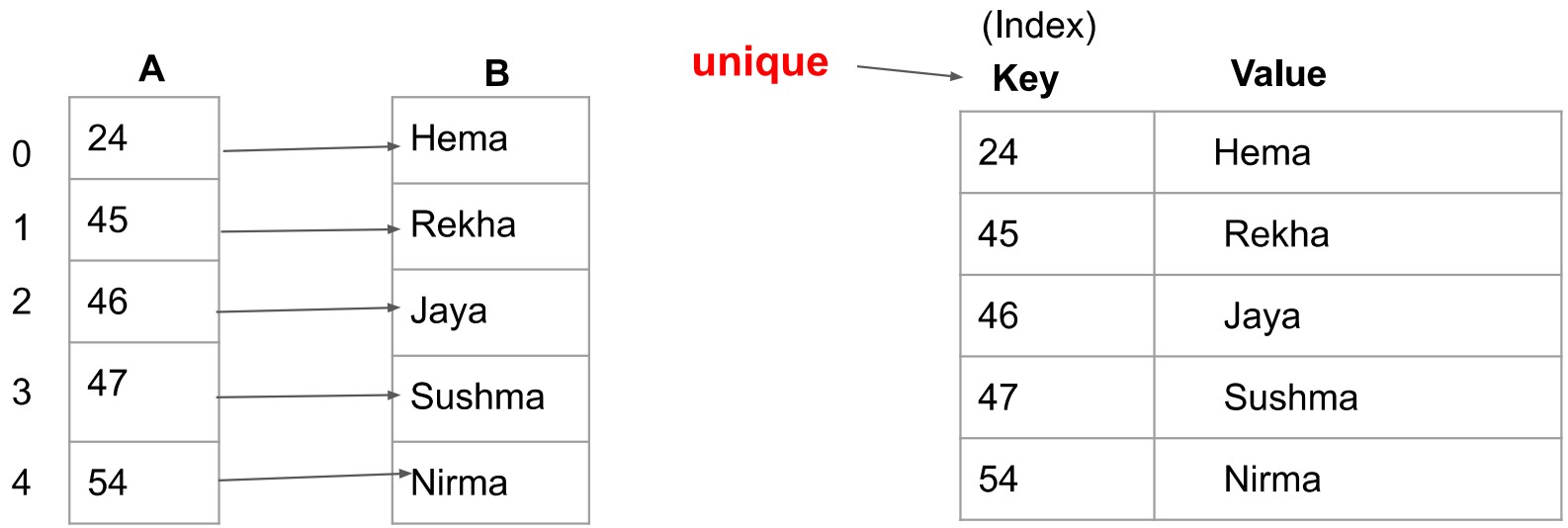
MAP

Maps are part of the **C++** STL (Standard Template Library). **Maps** are the associative containers that store sorted key-value pair, in which each key is unique and it can be inserted or deleted but cannot be altered. Values associated with keys can be changed.

Database --

Roll no -----> Student's Name

MAP sorted ----------> (Key-Value Pair)



QUESTION

Q.) Write a program to find out the frequency of all unique number in an array.

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
int main()
{
map<int,int> m;

int a[6]={1,3,2,1,2,1};

for(int i=0; i<6; i++)
{
        m[a[i]]++;
}
```

```
  cout << "Element  Frequency";
   cout << endl;
   for (auto i : m)
   {
        cout << i.first << "\t\t\t" << i.second ;
        cout<< endl;
  }
  return 0;
}
```

Output

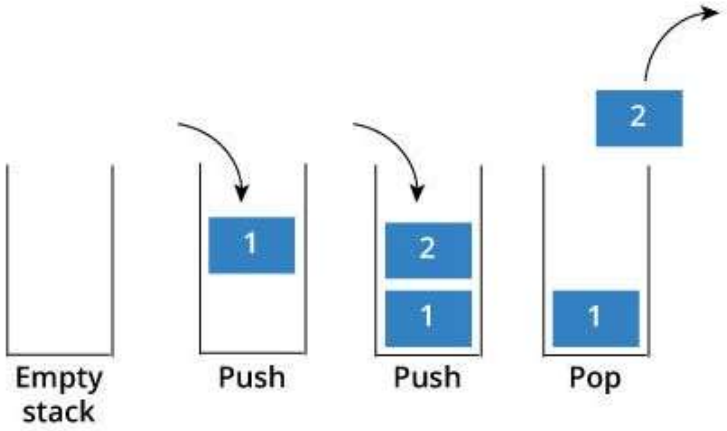
Element	Frequency
1	3
2	2
3	1

STACK

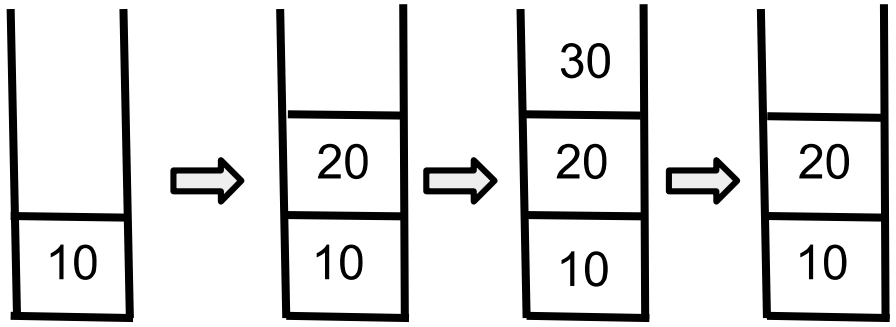
Stack

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be **LIFO**(Last In First Out) or **FILO**(First In Last Out).

There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen.



```
#include<bits/stdc++.h>
using namespace std;
int main()
{
  stack<int> s;
  s.push(10);
  s.push(20);
  s.push(30);
  s.pop();
  return 0;
}
```

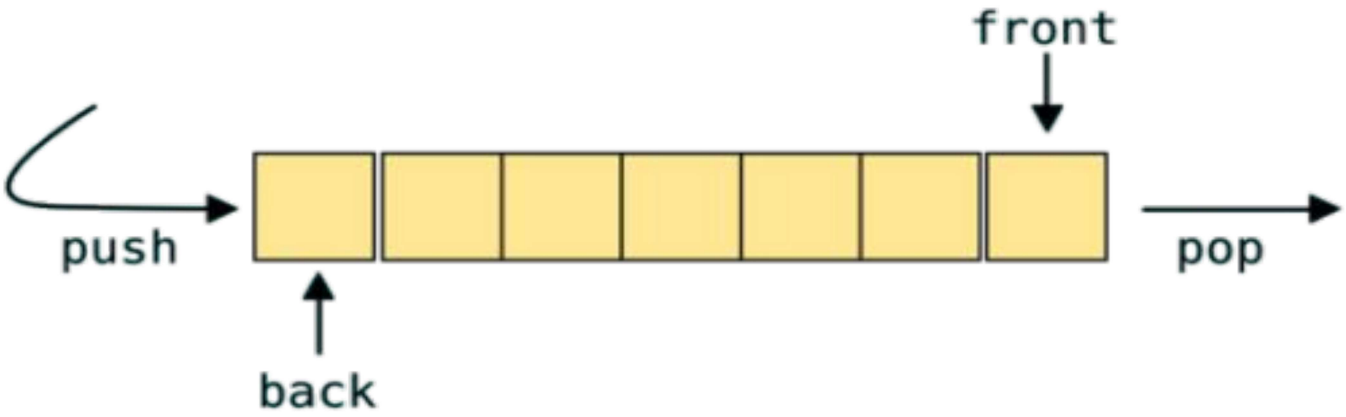


QUEUE

Queue

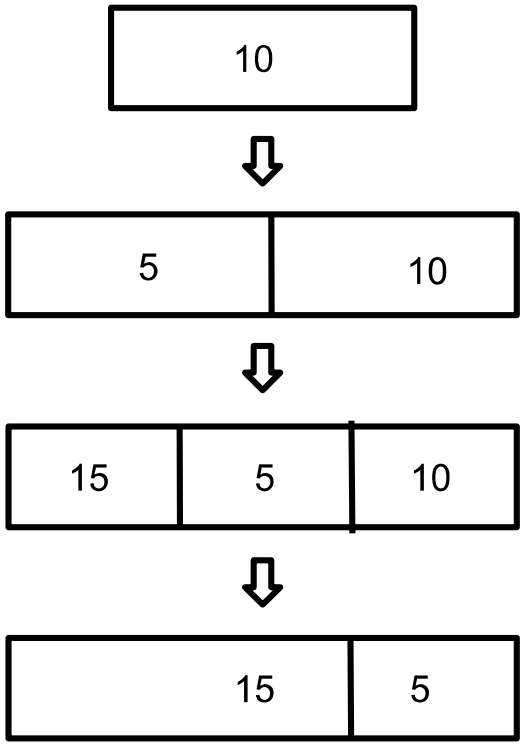
Queue is a data structure designed to operate in **FIFO** (First in First out) context. In queue elements are inserted from rear end and get removed from front end.

.



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

int main()
{
   queue<int> q;
   q.push(10);
   q.push(5);
   q.push(15);
   q.pop();
   return 0;
}
```



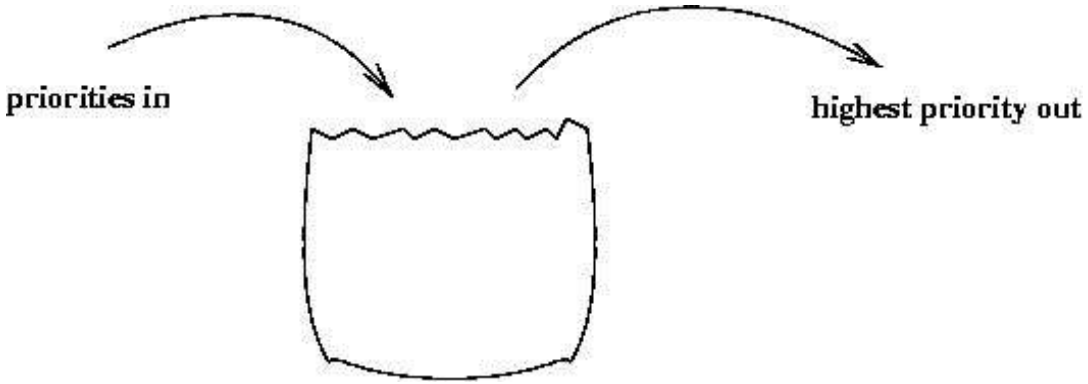
PRIORITY QUEUE

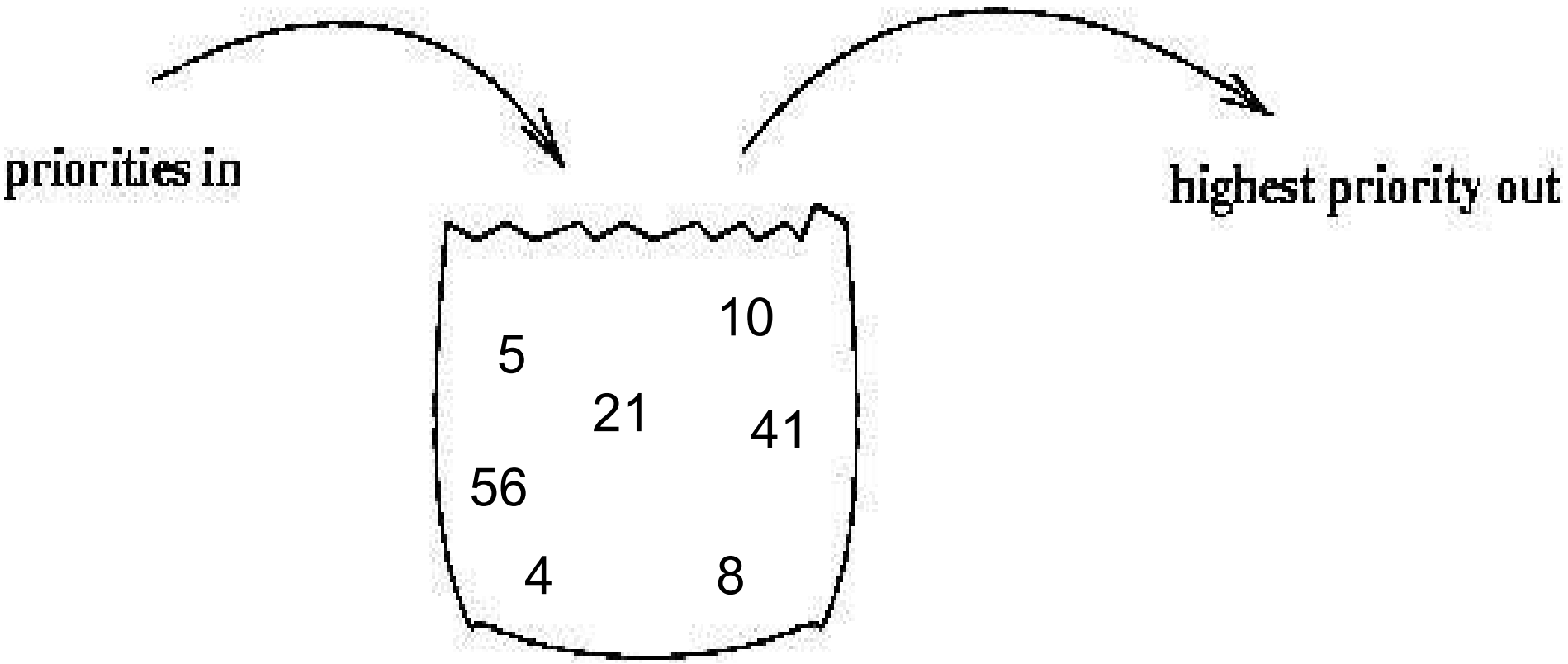
Priority Queue

Priority queue is a container which is designed such that the first element of the queue is greatest of all the elements and the elements are in descending order.

Declaration **- priority_queue<int> q;**

.




priorities in
highest priority out
10
5
21
41
56
4
8

SET

Set

Sets are a type of containers in which each element has to be **unique** because the value of the element identifies it. The values are stored in a specific order.

Properties:

1. No duplicates
2. Sorted in ascending order.
3. Searches an element in O(log n) time.
4. Unindexed.

We can compromise with the sorted order for improving the search time, by using unordered set, which has search time of O(1).

Insertion in set

Insert 5

5				

Insert 2

2	5			

Insert 4

2	4	5		

Insert 3

2	3	4	5	

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   set < int > s;
   s.insert( 60 ) ;
   s.insert( 10 ) ;
   s.insert( 20 ) ;
   s.insert( 20 ) ;
   s.insert( 40 ) ;
   s.erase(10);
  if( s.find(40)==s.end())
       cout<<"40 not found.";
   else
       cout<<"40 found.";
   cout<<endl;
   if(s.find(10)==s.end())
       cout<<"10 not found.";
   else
       cout<<"10 found.";
   return 0;
}
```

Output

40 found.
10 not found.

GCD OR HCF

GCD or HCF

GCD (Greatest Common Divisor) or HCF (Highest Common Factor) of n numbers is the largest number that divides all of them.

Program to find GCD or HCF of two numbers

36 = 2*2*3*3

60 = 2*2*3*5

GCD = Multiplication of common factors

= 2*2*3 = 12

Example of GCD

To find gcd in c++ we use __gcd() function.

To find gcd of two numbers a and b we can use
gcd = **__gcd(a, b)**

Eg : gcd of 4 and 8 is
Solution : 4

QUESTION

DIVISIBLE ARRAY

https://ideone.com/e2nfj7