```
//vector
//pair
//set



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


struct pr{
int v1,v2;
};


int main(){
int n=5;
cin>>n;
int a[n]; -> array static size
```

VECTORS

```
vector->dynamic size array
//vector <data_type> vector_name;
//vector_name.size()->vector size
vector <int> v;
cout<<v.size()<<"\n";

//vector <data_type> vector_name(vector_size,default_value);
vector <int> v1(n,80);
cout<<v1.size()<<"\n";
for(int i=0;i<n;i++)
cout<<v1[i]<<" ";
cout<<"\n";

//vector_name.resize(size,value); only newly added elements are
initialised with value
v1.resize(7,90);
cout<<v1.size()<<"\n";
for(int i=0;i<v1.size();i++)
cout<<v1[i]<<" ";
cout<<"\n";



//vector_name.assign(size,value); all elements are initialised with value
v1.assign(7,90);
cout<<v1.size()<<"\n";
for(int i=0;i<v1.size();i++)
cout<<v1[i]<<" ";
cout<<"\n";
```

```
//vector_name.push_back(value);
v1.push_back(500);
cout<<v1.size()<<"\n";

//vector_name.pop_back();
v1.pop_back();
cout<<v1.size()<<"\n";

// vector_name.begin() -> returns the starting pointer

// ith position pointer -> v.begin()+i;

// ending pointer -> v.end() -> v.begin()+v.size()

//v.empty()-> returns whether vector is empty

//v.insert(pos,value); -> pos represents iterator to the ith position


int value = 5;
int pos = 2;
v.insert(v.begin()+pos,value);


vector <pr> v; // pr is a structure
```

PAIRS

```
// pair <data1,data2> pair_name;

// first element pair_name.first

// second element access pair_name.second

pair <int,int> p1;
pair <float,int> p2;
pair <pair<int,int>,pair<float,int>> p;

// pair insert-> p1.first, p1.second
// p1 = make_pair(v1,v2);
// v1,v2,v3,v4
// p1 = make_pair(v1,v2);
// p2 = make_pair(v3,v4);
```

```
// p = make_pair(make_pair(v1,v2),make_pair(v3,v4));
// p = {{v1,v2},{v3,v4}};

// make_pair(v1,v2)=={v1,v2}

vector <pair<int,int>> vp;

p = {{3,4},{5,6}};
cout<<p.first.first<<" "<<p.first.second<<" "<<p.second.first<<"
"<<p.second.second<<"\n";
p.second.first = 10;
cout<<p.first.first<<" "<<p.first.second<<" "<<p.second.first<<"
"<<p.second.second<<"\n";
```

SETS

```
//set<data_type> set_name;
set <int> s;

//collection of unique elements sorted in some order -> default increasing

//set_name.insert(value); -> O(logn)
s.insert(2);
s.insert(3);
cout<<s.size()<<"\n";

//set_name.erase(value); or set_name.erase(iterator);->O(logn)
s.erase(3);
cout<<s.size()<<"\n";

//s.find(value)->return pointer to the value if value doesn't exist
s.find() -> O(logn)

//s.size()

//s.empty()

//set_name.count(value) -> 1 or 0 whether value exist or not  -> O(logn)

//set_name.find(value)==set_name.end()
```

```
auto ptr = s.find(2);
s.erase(ptr);
cout<<s.size()<<"\n";


return 0;
}
```