STL

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

stack<int> si;
si.push(10);
si.push(20);
if ( !si.empty() )
    cout << si.top();  // writes 20
si.pop();
cout << si.size();  //  writes 1
cout << si.top();  // writes 10
```

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

queue<int> qi;
qi.push(10);
qi.push(20);
if ( !qi.empty() )
    cout << qi.front();  // writes 10
cout << qi.back();  // writes 20
qi.pop();
cout << qi.size();  //  writes 1
cout << qi.front();  //  writes 20
```

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

vector<int> vi;
vi.push_back(10);
vi.push_back(20);
vi.push_back(30);
cout << vi.size();  // writes 3
cout << vi.front(); // writes 10
cout << vi.back();  // writes 30
vi[1] = 40;
//  vi[3] = 50; would be undefined behavior
for (size_t k = 0; k < vi.size(); k++)
    cout << vi[k] << endl;
 // writes 10 40 30, one per line
vi.pop_back();
for (size_t k = 0; k < vi.size(); k++)
    cout << vi[k] << endl;
 // writes 10 40, one per line
vi.at(1) = 60;
vi.at(3) = 70;  // throws exception
vector<double> vd(10);
// vd.size() is 10, each element is 0.0
vector<string> vs(10, "Hello");
// vs.size() is 10, each element is "Hello"
int a[5] = { 10, 20, 30, 40, 50 };
vector<int> vx(a, a+5);
// vx.size() is 5, vx[0] is 10, vx[1] is
//   20, ..., vx[4] is 50
```

```
vector<int> vi;
vi[0] = 10;  // undefined!  vi.size() is 0
   // there are no elements
```

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

list<int> li;
li.push_back(20);
li.push_back(30);
li.push_front(10);
cout << li.size();  // writes 3
cout << li.front(); // writes 10
cout << li.back();  // writes 30
li.push_front(40);
li.pop_front();

  li.begin()                   li.end()
     v                            v
    10         20         30

for (list<int>::iterator p = li.begin();
                        p != li.end(); p++)
   cout << *p << endl;
 // writes 10 20 30, one per line
```

```
list<double> ld(10);
// ld.size() is 10, each element is 0.0
list<string> ls(10, "Hello");
// ls.size() is 10, each element is "Hello"
vector<string> vs(ls.begin(), ls.end());
// vs.size() is 10, vs[0] is "Hello", vs[1]
//   is "Hello", ..., vs[9] is "Hello"

list<int>::iterator p = li.end();
p--;
p--;
//  p -= 2 won't compile

 li.begin()    p              li.end()
     v         v                 v
    10         20         30

list<int>::iterator q = li.insert(p, 40);

 li.begin()    q        p         li.end()
     v         v        v                v
    10         40       20         30
```

```
list<int>::iterator p;
...

 li.begin()    p             li.end()
     V         v                 v
    10        20        30

list<int>::iterator q = li.erase(p);

 li.begin()              q    li.end()
     V                   v        v
    10                  30

It's now undefined to use p (*p, p++, etc.)
until you assign p a new value
```

```
vector<int> vi;
...
vector<int>::iterator p = vi.end() - 2;

 vi.begin()    p             vi.end()
     V         v                 v
    10        20        30

vector<int>::iterator q = vi.insert(p, 40);

 vi.begin()    q                  vi.end()
     V         v                        v
    10        40       20        30

It's now undefined to use p (*p, p++, etc.)
until you assign p a new value

p = vi.erase(q);

 vi.begin()    p           vi.end()
     V         v               v
    10        20       30

It's now undefined to use q (*q, q++, etc.)
until you assign q a new value
```

```
int* find(int* b, int* e, const int& target)
{
    for ( ; b != e; b++)
        if (*b == target)
            break;
    return b;
}

int main()
{
    int a[5] = { 10, 50, 40, 20, 30 };
    int k;
    cin >> k;
    int* p = find(a, a+5, k);
    if (p == a+5)
        ... not found ...
    else
        ... found, p points to the first element with that value
}
```

```
template<typename T>
T* find(T* b, T* e, const T& target)
{
    for ( ; b != e; b++)
        if (*b == target)
            break;
    return b;
}

int main()
{
    int a[5] = { 10, 50, 40, 20, 30 };
    int k;
    cin >> k;
    int* p = find(a, a+5, k);
    if (p == a+5)
        ... not found ...
    else
        ... found, p points to the first element with that value

    string sa[4] = { "Lucy", "Ricky", "Fred", "Ethel" };
    string* sp = find(sa, sa+4, “Fred”);
    ...
}
```

```
template<typename Iter, typename T>
Iter find(Iter b, Iter e, const T& target)
{
    for ( ; b != e; b++)
        if (*b == target)
            break;
    return b;
}

int main()
{
    int a[5] = { 10, 50, 40, 20, 30 };
    ...
    int* p = find(a, a+5, k);
    if (p == a+5)
        ... not found ...
    else
        ... found, p points to the first element with that value

    list<string> ls;
    ...
    list<string>::iterator q = find(ls.begin(), ls.end(), "Fred");
    if (q == ls.end())
        ... not found ...
    ...

    vector<int> vi;
    ...
    vector<int>::iterator r = find(vi.begin(), vi.begin()+5, 42);
    if (r == vi.begin()+5)
        ... not found ...
}
```

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

int main()
{
    vector<int> vi;
    ...
    vector<int>::iterator p = find(vi.begin(), vi.end(), 42);
    if (p != vi.end())
    {
        int n = count(vi.begin(), vi.end(), 0);
        reverse(vi.begin(), vi.end());
        ...
    }
    sort(vi.begin(), vi.end());
    ...
}
```

```
template<typename Iter>
Iter findFirstNegative(Iter b, Iter e)
{
    for ( ; b != e; b++)
        if (*b < 0)
            break;
    return b;
}

template<typename Iter>
Iter findFirstEmpty(Iter b, Iter e)
{
    for ( ; b != e; b++)
        if (b->empty())
            break;
    return b;
}
```

```
template<typename Iter, typename Func>
Iter find_if(Iter b, Iter e, Func f)
{
    for ( ; b != e; b++)
        if (f(*b))
            break;
    return b;
}

bool isNegative(int k)
{
    return k < 0;
}

bool isEmpty(string s)
{
    return s.empty();
}

int main()
{
    vector<int> vi;
    ...
    vector<int>::iterator p = find_if(vi.begin(), vi.end(), isNegative);
    if (p == vi.end())
        ... not found ...

    list<string> ls;
    ...
    list<string>::iterator q = find_if(ls.begin(), ls.end(), isEmpty);
    ...
}
```

```
bool isGreater(int i, int j)
{
    return i > j;
}

bool makesLessThan(const Employee& e1, const Employee& e2)
{
    return e1.salary() < e2.salary();
}

bool hasBetterRecord(const Team& t1, const Team& t2)
{
    if (t1.wins() > t2.wins())
        return true;
    if (t1.wins() < t2.wins())
        return false;
    return t1.ties() > t2.ties();
}

int main()
{
    vector<int> vi;
    ...
    sort(vi.begin(), vi.end(), isGreater);
    ...
    Employee ea[100];
    ...
    sort(ea, ea+100, makesLessThan);
    ...
    vector<Team> league;
    ...
    sort(league.begin(), league.end(), hasBetterRecord);
    ...
}
```