```
/************** MAX, MIN VALUE *************/
#include<climits>
INT_MAX = ~(1 << 31)   // for 64bit machine
INT_MIN = 1 << 31      // for 64bit machine
UINT_MAX = (uint)(~0)  // 32 bit all equal 1
LONG_MAX
LONG_MIN
ULONG_MAX
```

```
/************************* rand / srand *****************************/
#include<stdlib.h>
#include<time.h>
srand((unsigned)time(0));  // initialize rand seed
rand() % len + a;          // [a, len + a)
rand()/double(RAND_MAX);   // random floating number from [0.0, 1.0]
```

```
/********************** rotate ***************************/
rotate (Iterator first, Iterator middle, Iterator last);
// Rotates elements in the range [first,last),
// the element pointed by middle becomes the new first element.
Example:
for (int i=1; i<10; ++i) vec.push_back(i);  // 1 2 3 4 5 6 7 8 9
rotate(vec.begin(),vec.begin()+3,vec.end());// 4 5 6 7 8 9 1 2 3
```

```
/********************** string **************************/
string(char[] chArr) or string(char* chArr);  // string constructor
string(int n, char ch);                        // string constructor with n characters of ch

string str = "1234";
str[i];                                        // access i th character
str.size();  or str.length();
str.substr(start);                             // [start,  )
str.substr(start, length);                     // [start, start + length - 1]       str doesn't change

str.append("abc"); or str+="abc";
str.append(1, 'a');                           // append character
size_t found = str.find("ab");                // return pos where "ab" first occur in str.
if (found!=string::npos)  cout << "found";    //

str.erase(2);                                 // erase substring starting from 2.    [2, ) str = "12"
str.erase(pos, length);                       // erase length characters starting from pos
str.insert(2, "sz");                          // insert characters starting from pos 2.
str.replace(pos, len, "newStr");              // replace substring starting from pos with length = len as "newStr"

str1.compare(str2);                           // 0 equal; -1 str1 comes first in lexicographic order
reverse(str.begin(), str.end());        // reverse string.       str changes!!!!!!!!!!  no return value
```

```
/********* math *********/
#include <math.h>
M_PI
cos(theta * M_PI / 180.0)
acos(-1) = M_PI;
sqrt()
round()
pow(n, k);
```

```
/********** Node **********/
class ListNode{
public:
    int val;
    ListNode* next;
    ListNode(int val){
        this->val = val;
        this->next = NULL;
    }
};

class TreeNode{
public:
    int val;
    TreeNode *left;
    TreeNode *right;
    TreeNode(int val){
        this->val = val;
        this->left = NULL;
        this->right = NULL;
    }
};
```

```
/********************** vector **************************/
#include <vector>
vector<int> v;
vector<int> v(size, 0);   // create a vector with length of size and initialize all elements to 0;
vector<vector<int>> v(N, vector<int>(M, 0)); //initialize N * M 2d vector to zero

int val = v[i];           // random access
v.empty();                // return bool to indicate empty or not
v.push_back(e);           // insert element to end
v.pop_back();             // delete last element
v.clear();
v.front();                // return first element
v.back();                // return last element
v.erase(v.begin() + 5);   // delete 6th element;
v.insert(v.begin(), var)  // insert var in first position
v.begin();                // return iterator pointing to first element;
v.end();                  // return iterator pointing to null behind last element;
v.resize(num);                     // resize the length of vector
v.resize(num, val);                // resize vector by using val to padding  (default is 0);
                                   // resize(num, val) can be used for constructor in class;

#include <algorithm>
sort(v.begin(), v.end());          // sort vector and from min to max by default

struct cmp{
    bool operator() (int x, int y){
        return x > y;              // descending order
    }
} cmpObj;
sort(v.begin(), v.end(), cmpObj);  // sort with self-defined comparator
```

```
/********************* unordered_set, set *************************/
unordered_set<int> Set;
Set.insert(val);
Set.erase(val);
Set.erase(iterator);
Set.size();
Set.empty();
if(Set.find(1) != Set.end()) cout << "found" << endl;   // find val
for(iter = Set.begin(); iter != Set.end(); ++iter)      // traverse
    cout << *iter <<endl;

struct cmp{
    bool operator()(Node* a, Node* b){
        return (a->val) < (b->val);
    }
};
set<Node*, cmp> s;           //intialize ordered set with comparator
```

```
/********************* array *************************/
int nums[10] = {0};
[array to vector]  vector<int> vec(&nums[0], &nums[10]);
```

```
/****************** string, char, integer conversion ***************/
[int to string]           to_string(num);
[string to int]           stoi(s);          // i.e.  int val = stoi("1024");
[char to string]          string(1, ch);
[charr array to string] string(charArr);
```

```
/********************* unordered_map, map *************************/
#include<unordered_map>
unordered_map<int, string> Map;                          // O(1) time complexity
#include<map>
map<int, string> treeMap;                                // O(LogN) time complexity

Map[1] = "one";                                          // insert
string str = Map[1];                                     // get
if(Map.find(1) != Map.end()) cout << Map[1] << endl;     // search key
Map.erase(1);                                            // delete

unordered_map<int, string>::iterator it = Map.find(1);   // find by key
if(iter != Map.end()) cout << iter->second;
else cout << "not found";

for(auto iter : Map)
    cout << iter->first << iter->second <<end;           // traverse

Map.erase("one");                                        // delete
Map.empty();
Map.size();
```

```
/**** Priority queue ****/
#include <priority_queue>
priority_queue<int> pq;
pq.push(val);
pq.top();
pq.pop();

struct cmp{
  bool operator()(Node* a, Node* b){
    return a -> x > b -> x;          // build min heap
  }
};
priority_queue<Node*, vector<Node*>, cmp> pq;
pq.push(new Node(1, 2));
```

```
/**** stack ****/
#include <stack>
stack<int> s;
s.top();
s.push();
s.pop();
```

```
/**** queue ****/
#include <queue>
queue<int> q;
q.front();
q.back();
q.push();
q.pop();
q.empty();
```

```
/**** deque ****/
#include<deque>
deque<int> dq;
dq.push_back(val);
dq.push_front(val);
dq.pop_back();
dq.pop_front();
```

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

// To execute C++, please define "int main()"
int main() {
  return 0;
}
```

```
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
```