Math functions:

Function	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x
asin(x)	Returns the arcsine of x
atan(x)	Returns the arctangent of x
cbrt(x)	Returns the cube root of x
ceil(x)	Returns the value of x rounded up to its nearest integer
cos(x)	Returns the cosine of x
cosh(x)	Returns the hyperbolic cosine of x
exp(x)	Returns the value of Ex
expm1(x)	Returns ex -1
fabs(x)	Returns the absolute value of a floating x
fdim(x, y)	Returns the positive difference between x and y
floor(x)	Returns the value of x rounded down to its nearest integer
hypot(x, y)	Returns sqrt(x2 +y2) without intermediate overflow or underflow
fma(x, y, z)	Returns x*y+z without losing precision
fmax(x, y)	Returns the highest value of a floating x and y
fmin(x, y)	Returns the lowest value of a floating x and y
fmod(x, y)	Returns the floating point remainder of x/y
pow(x, y)	Returns the value of x to the power of y
sin(x)	Returns the sine of x (x is in radians)
sinh(x)	Returns the hyperbolic sine of a double value
tan(x)	Returns the tangent of an angle
tanh(x)	Returns the hyperbolic tangent of a double value

Vectors:

push_back(x)	Add number to the end
at(index)	Acces element, this is saferthan [ ]
pop_back()	Delete last element
size()	returns the number of elements present in the vector
clear()	removes all the elements of the vector
front()	returns the first element of the vector
back()	returns the last element of the vector
empty()	returns **1** (true) if the vector is empty
capacity()	check the overall size of a vector

using vector iterator in for loop:

```
vector<int>::iterator iter;
for (iter = num.begin(); iter != num.end(); ++iter) {
  cout << *iter << "  ";
}
```

Maps:

begin() Returns an iterator to the first element in the map.

end() Returns an iterator to the theoretical element that follows the last element in the map.

size() Returns the number of elements in the map.

max_size() Returns the maximum number of elements that the map can hold.

empty() Returns whether the map is empty.

pair insert(keyvalue, mapvalue) Adds a new element to the map.

erase(iterator position) Removes the element at the position pointed by the iterator.

erase(const g) Removes the key-value ‘g’ from the map.

clear() Removes all the elements from the map.

String:

1. enc.erase(remove(enc.begin(), enc.end(), ' '), enc.end());
2.

int compare(const string& str)	It is used to compare two string objects.
int length()	It is used to find the length of the string.
void swap(string& str)	It is used to swap the values of two string objects.

string substr(int pos,int n)	It creates a new string object of n characters.
int size()	It returns the length of the string in terms of bytes.
void resize(int n)	It is used to resize the length of the string up to n characters.
string& replace(int pos,int len,string& str)	It replaces portion of the string that begins at character position pos and spans len characters.
string& append(const string& str)	It adds new characters at the end of another string object.
char& at(int pos)	It is used to access an individual character at specified position pos.
int find(string& str,int pos,int n)	It is used to find the string specified in the parameter.
int find_first_of(string& str,int pos,int n)	It is used to find the first occurrence of the specified sequence.
int find_first_not_of(string& str,int pos,int n)	It is used to search the string for the first character that does not match with any of the characters specified in the string.
int find_last_of(string& str,int pos,int n)	It is used to search the string for the last character of specified sequence.
int find_last_not_of(string& str,int pos)	It searches for the last character that does not match with the specified sequence.
string& insert()	It inserts a new character before the character indicated by the position pos.
int max_size()	It finds the maximum length of the string.
void push_back(char ch)	It adds a new character ch at the end of the string.
void pop_back()	It removes a last character of the string.

string& assign()	It assigns new value to the string.

Tree:

Definition for a binary tree node.
struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
};

1. **sort(first_iterator, last_iterator)** – To sort the given vector.
2. **sort(first_iterator, last_iterator, greater<int>())** – To sort the given container/vector in descending order
3. **reverse(first_iterator, last_iterator)** – To reverse a vector. (if ascending -> descending OR if descending -> ascending)
4. ***max_element (first_iterator, last_iterator)** – To find the maximum element of a vector.
5. ***min_element (first_iterator, last_iterator)** – To find the minimum element of a vector.
6. **accumulate(first_iterator, last_iterator, initial value of sum)** – Does the summation of vector elements
7. **count(first_iterator, last_iterator,x)** – To count the occurrences of x in vector.
8. **find(first_iterator, last_iterator, x)** – Returns an iterator to the first occurrence of x in vector and points to last address of vector ((name_of_vector).end()) if element is not present in vector.
9. **binary_search(first_iterator, last_iterator, x)** – Tests whether x exists in sorted vector or not.
10. **lower_bound(first_iterator, last_iterator, x)** – returns an iterator pointing to the first element in the range [first,last) which has a value not less than ‘x’.

11. **upper_bound(first_iterator, last_iterator, x)** – returns an iterator pointing to the first element in the range [first,last) which has a value greater than ‘x’.