string-int:#include <string>#include<sstream>int num =stoi(str);string str2= to_string(num2);

string-double:double num = stod(str);

type-casting:double Double = static_cast<double>(Int);

inputbuffer:cin.ignore();cin.clear();

set: #include<set> set <dt> a; a.insert(s[i]); mySet.erase(10); auto it = mySet.find(5);

for (auto it = mySet.begin(); it != mySet.end(); ++it) {std::cout << *it << " ";}//iterate set

for (const auto& elem : mySet) { std::cout << elem << " ";} *set sort automatically

auto maxElementIt = mySet.rbegin(); int maxElement = *maxElementIt;→max element

auto minElementIt = mySet.begin(); int minElement = *minElementIt;→min element
gcd:__gcd(a,b);int maxGCD = n / 2;**lcm:** (a * b) / gcd(a, b)

vector: #include <vector>vector<int> myVector; myVector.push_back(42); myVector.pop_back();myVector.clear();copyVector(anotherVector); sort(vec.begin(), vec.end());

reverse(vec.begin(), vec.end());auto maxElement = max_element(vec.begin(), vec.end());

auto minElement = std::min_element(vec.begin(), vec.end());

array: auto maxElement = std::max_element(std::begin(arr), std::end(arr)); *maxElement;

auto minElement = std::min_element(std::begin(arr), std::end(arr)); *minElement;

int* maxElement = std::max_element(arr, arr + n); int* minElement = std::min_element(arr, arr + n);

string: #include <cctype> transform(str.begin(), str.end(), str.begin(), ::tolower);

#include <cctype> transform(str.begin(), str.end(), str.begin(), ::toupper);

Permutation: sort(vec.begin(), vec.end());

```
do {
    for (int num : vec) {
        std::cout << num << " ";
    }std::cout << std::endl;
} while (std::next_permutation(vec.begin(), vec.end()));
```

Subarray: for (int i = 0; i < arr.size(); ++i) {

for (int j = i; j < arr.size(); ++j) {

for (int k = i; k <= j; ++k) {

std::cout << arr[k] << " ";

}std::cout << std::endl;

}

}

Sieve of eranthonses: vector<int> sieveOfEratosthenes(int n) {

std::vector<bool> isPrime(n + 1, true);

std::vector<int> primes;

for (int p = 2; p * p <= n; ++p) {

if (isPrime[p]) {

for (int i = p * p; i <= n; i += p) {

isPrime[i] = false;

}

}

}

for (int p = 2; p <= n; ++p) {

if (isPrime[p]) {

primes.push_back(p);

}

}

return primes;

}

Prime: bool isPrime(int n) {

if (n <= 1)

return false;

```
    if (n <= 3)
        return true;
    // Check divisibility by all numbers from 2 up to the square root of n
    for (int i = 2; i <= sqrt(n); ++i) {
        if (n % i == 0)
            return false;
    }
    return true;
}
```

Subsequence:

```
while (i < n && j < m)
    {
        if (a[i] == b[j])
            j++;
        i++;
    } //j ==m for ss
```

Greedy:

```
vector<int> greedyCoinChange(int amount, const vector<int>& coins) {
    vector<int> result;
    // Sort coins in descending order
    vector<int> sortedCoins = coins;
    sort(sortedCoins.rbegin(), sortedCoins.rend());
    // Iterate through each coin denomination
    for (int coin : sortedCoins) {
        while (amount >= coin) {
            result.push_back(coin);
            amount -= coin;
        }
    }
```

```
    return result;
}
```

Dp: int fibonacciMemo(int n, vector<int>& memo) {

```
    if (n <= 1)
        return n;
    if (memo[n] != -1)
        return memo[n];
    memo[n] = fibonacciMemo(n - 1, memo) + fibonacciMemo(n - 2, memo);
    return memo[n];
```

logical op: unsigned int a = 0b1010, b = 0b1100; std::cout << "AND: " << (a & b) << ", OR: " << (a | b) << ", XOR: " << (a ^ b) << ", NOT (a): " << ~a << ", NOT (b): " << ~b << ", XNOR: " << (~(a ^ b)) ;

bitwisw: cout << "AND: " << (a & b) << ", OR: " << (a | b) << ", XOR: " << (a ^ b) << ", NOT (a): " << ~a << ", NOT (b): " << ~b ;

unsigned int num = 10;int shift = unsigned int rightShiftResult = num >> shift;

string leftShiftResult = std::bitset<sizeof(unsigned int)*8>(num << shift).to_string();

binaryconv: #include <bitset>int num = 10; // Example integer

```
    std::bitset<sizeof(int) * 8> binary(num); // Convert integer to binary
    // Store binary representation in a string
    std::string binaryString = binary.to_string();
```

pairvector:

```
bool compare(const pair<int, int> &a, const pair<int, int> &b)
{
    if (a.first != b.first)
        return a.first > b.first;
    else
        return a.second < b.second;
}
```

```
int main()
{
    int n, m;
    cin >> n >> m;
    vector<int> a(n), b(n);
    for (int i = 0; i < n; i++)
    {
        cin >> a[i];
    }
    for (int i = 0; i < n; i++)
    {
        cin >> b[i];
    }
    vector<pair<int, int> > c;
    for (int i = 0; i < n; i++)
    {
        c.push_back(make_pair(a[i], b[i]));
    }
    sort(c.begin(), c.end(), compare);
    for (int i = 0; i < m; i++)
    {
        cout << c[i].second << endl;
    }
    return 0;
}
```