{

    "": {

        "prefix": "evibe",

        "body": [

          "#include<bits/stdc++.h>",

          "",

          "#include<ext/pb\_ds/assoc\_container.hpp>",

          "#include<ext/pb\_ds/tree\_policy.hpp>",

          "",

          "using namespace std;",

          "using namespace \_\_gnu\_pbds;",

          "",

          "#define ll long long",

          "#define ull unsigned long long",

          "#define pq priority\_queue",

          "#define pb push\_back",

          "",

          "typedef tree<int, null\_type, less<int>, rb\_tree\_tag, tree\_order\_statistics\_node\_update> pbds; // find\_by\_order, order\_of\_key --> these will return iterator ",

          "//--> to get value --> \*a.find\_by\_order(i); \*a.order\_of\_key(X);",

          "//\*a.find\_by\_order(i); --> finding kth element ",

          "//\*a.order\_of\_key(i); --> finding number of elements smaller than X",

          "//\*a.lower\_bound(X); --> lower bound -> Lower Bound of X = first element >= X in the set",

          "//\*a.upper\_bound(X); --> Upper bound -> Upper Bound of X = first element > X in the set",

          "//a.erase(X); --> Remove X from the ordered set",

          "",

          "/\*",

          "//typedef tree<int, null\_type, less<int>, rb\_tree\_tag, tree\_order\_statistics\_node\_update> pbds; //this will sort the set in ascending order",

          "//typedef tree<int, null\_type, greater<int>, rb\_tree\_tag, tree\_order\_statistics\_node\_update> pbds; //this will sort the set in descending order",

          "//typedef tree<int, null\_type, less\_equal<int>, rb\_tree\_tag, tree\_order\_statistics\_node\_update> pbds; //this will sort the set in ascending order + equal values will be included",

          "\*/",

          "",

          "",

          "/\*",

          "# deque //both",

          "# stack //LIFO",

          "# queue //FIFO",

          "\*/",

          "",

          "//",

          "/\*",

          "# map2.insert(map1.begin(),map1.end()); //elements of map1 will be inserted to map2",

          "# map <char,int>::iterator it; ",

          "# it = mp.find('b');",

          "# a.erase(a.begin(), a.end())",

          "# a.erase('c'); //erases element mapped at 'c'.",

          "# if(mp.empty()) {cout<<\"Map is empty\"<<endl;} //returns true or false value",

          "# for(it=a.begin(); it!=a.end(); it++) {cout<<it->first<<\" \"<<it->second<<endl;}",

          "# map1.swap(map2); //swap contents of 2 maps namely map1 and map2.   ",

          "\*/",

          "",

          "//",

          "using vb = vector<bool>;",

          "using vvb = vector<vb>;",

          "using vi = vector<int>;",

          "using vvi = vector<vi>;",

          "using vl = vector<ll>;",

          "using vvl = vector<vl>;",

          "using vc = vector<char>;",

          "using vvc = vector<vc>;",

          "using vs = vector<string>;",

          "",

          "//",

          "#define all(a)           a.begin(), a.end()",

          "#define vsort(a)         sort(a.begin(), a.end())",

          "#define srev(a)          reverse(a.begin(), a.end())",

          "#define ssort(a)         sort(a.begin(), a.end())",

          "#define grtsrt(v)        sort(v.begin(), v.end(), greater<ll>())",

          "#define vrot(a, rot\_by)  rotate(a.begin(), a.end() - rot\_by, a.end())",

          "//",

          "#define vmin(a)          \*min\_element(a.begin(), a.end())",

          "#define vmax(a)          \*max\_element(a.begin(), a.end())",

          "#define smax(a)          \*(a.rbegin()) //1st check this condition --> if (!a.empty())  ",

          "#define smin(a)          \*a.begin() //1st check this condition --> if (!a.empty())",

          "//",

          "#define vcount(v, a)     count(v.begin(), v.end(), a)",

          "#define scount(v, a)     count(v.begin(), v.end(), a)",

          "//",

          "#define mp(a, b)         make\_pair(a, b)",

          "//",

          "#define ub\_pos(a, x)     upper\_bound(a.begin(), a.end(), x) - a.begin()",

          "#define lb\_pos(a, x)     lower\_bound(a.begin(), a.end(), x) - a.begin()",

          "#define bins(a, n)       binary\_search(a.begin(), a.end(), n) //returns true or false value",

          "//",

          "#define lower\_case(a)    transform(a.begin(), a.end(), a.begin(), ::tolower)",

          "#define upper\_case(a)    transform(a.begin(), a.end(), a.begin(), ::toupper)",

          "//",

          "#define mem(a, x)        memset(a, x, sizeof(a))",

          "//",

          "#define nxp(v)           next\_permutation(v.begin(), v.end()); //generates all possible permutations",

          "//",

          "#define toint(a)         atoi(a.c\_str())",

          "#define total\_sum(a)     accumulate(a.begin(), a.end(), 0) //total\_sum initializing with 0; //this fnc gives the total sum of all numbers in the array",

          "//",

          "#define check(n, pos)    (n & (1<<(pos)))",

          "#define biton(n, pos)    (n | (1<<(pos)))",

          "#define bitoff(n, pos)   (n & ~(1<<(pos)))",

          "",

          "#define get\_bit(n, pos)  (bool)(n & (1 << pos)) //returns bool value --> 0 or 1;",

          "//indexing starts from 0 --> 1st position bit is 0th bit, then 1st bit, 2nd bit and so on",

          "#define set\_bit(n, pos)  (n | (1 << pos))",

          "//indexing starts form 0 --> meaning k=2; this 2nd bit is the (k-1)th; 1st position bit",

          "#define clear\_bit(n, pos)  (n & (~(1 << pos))) //~(1 << k) means the kth bit of ~(1 << k) is always 0",

          "//clearing the kth bit --> ensuring that the kth bit is unset (0)",

          "#define update\_bit(n, pos, value) ((clear\_bit(n, pos)) | (value << pos))",

          "//value represents the bit we want to set at the kth position. To do so we need to clear the kth bit",

          "//so we declared it first, then we are going to set the bit and the operation between them is or",

          "",

          "const double pi= 3.141592653589793238462643383279502884197169399375105820974944;",

          "const ll N=1e9+7;",

          "const ll mod = 1e9 + 7, inf = 1e18;",

          "",

          "/\*\*/",

          "//if(s1.compare(s2) == 0) {cout << s << \" is equal to \" << s1 << endl;} --> here s1 and s2 are 2 strings and we are comparing these strings; if == 0 than they are same otherwise different",

          "//string s = \"dog:cat\";  int pos = s.find(\":\");  string sub = s.substr(pos + 1);   // Copy substring after pos; -->  String is: cat",

          "//string s1 = \"Geeks\";   string r = s1.substr(3, 2);   // Copy two characters of s1 (starting from position 3); -->  String is: ks",

          "",

          "",

          "/\*\*/",

          "//\_\_int128(n)\*mid\*mid\*4 <-- Here n\*mid\*mid\*4 is written using --> \_\_int128() which may not support in 64 bit computer ",

          "//but this is used for calculating large int value that can't be processed in i64 or long long",

          "",

          "",

          "/\*\*/",

          "//2^n is equal to (1 << n)",

          "//xor of two same numbers is equal to 0",

          "",

          "",

          "/\*",

          "",

          "//a+b=a|b + a&b   //a+b=a⊕b+2(a&b) here (a | b) equals to (a⊕b + a&b)",

          "//a-b=a-(a&b)-x where x is basically (bitwise not of a) & b",

          "",

          "//a^(a&b) = (a|b)^b",

          "//b^(a&b) = (a|b)^a",

          "",

          "",

          "Some properties of bitwise operations:",

          "a|b = a⊕b + a&b",

          "a⊕(a&b) = (a|b)⊕b",

          "b⊕(a&b) = (a|b)⊕a",

          "(a&b)⊕(a|b) = a⊕b",

          "",

          "",

          "Addition:",

          "a+b = a|b + a&b",

          "a+b = a⊕b + 2(a&b)",

          "",

          "",

          "Subtraction:",

          "a-b = (a⊕(a&b))-((a|b)⊕a)",

          "a-b = ((a|b)⊕b)-((a|b)⊕a)",

          "a-b = (a⊕(a&b))-(b⊕(a&b))",

          "a-b = ((a|b)⊕b)-(b⊕(a&b))",

          "",

          "\*/",

          "",

          "",

          "",

          "/\*",

          "",

          "int rotate\_by=k%(n+1);",

          "rotate(a.begin(), a.end() - rotate\_by, a.end());",

          "",

          "\*/",

          "",

          "",

          "/\*",

          "",

          "# bitset<66> bits(x); //bitset operation --> converts any number to bits, can mention how many bits we want; here 66 bits that we want",

          "//using the following operation we can convert the bits into string:",

          "",

          "    string res=\"\";",

          "    res=bits.to\_string();",

          "",

          "//using the following operation we can exclude the extra 0's that comes in front of the number:",

          "",

          "    reverse(res.begin(), res.end());",

          "    while(res.back()=='0')",

          "    {",

          "        res.pop\_back();",

          "    }",

          "    reverse(res.begin(), res.end());",

          "",

          "here if 4 -> 000000100  is found using bitset; by using the above method we can get 100 the required bits only.",

          "",

          "\*/",

          "",

          "",

          "",

          "void solve()",

          "{",

          "}",

          "",

          "",

          "int main()",

          "{",

          "    ios\_base::sync\_with\_stdio(0);",

          "    cin.tie(0); ",

          "    cout.tie(0);",

          "",

          "    // #ifndef ONLINE\_JUDGE",

          "    //     freopen(\"input.txt\", \"r\", stdin);",

          "    //     freopen(\"output.txt\", \"w\", stdout);",

          "    // #endif",

          "",

          "    int t=1;",

          "    cin>>t;",

          "",

          "    for(int i=0; i<t; i++)",

          "    {",

          "        solve();",

          "    }",

          "",

          "    return 0;",

          "}"

        ],

        "description": ""

      }

}