1. Server cost reduction
2. Subarray having odd number of divisor
3. Max discounts
4. Backspace string compare
5. **Binary manipulation X2**
6. Trilogy
7. **Quiz competition X2**
8. **Do they belong X2**
9. **Re arrange student**
10. TTI cache
11. **String subsequence**
12. **Get the group X2**
13. **Best sum any tree path X3**
14. Whole minute dilemma
15. Modest number
16. **Valid BST permutation**

1.Do they belong ?

2.Server Cost Reduction

```
long long solve(int N, vector<int> from, vector<int> to, vector<int> weight, int k){ //C++20
    vector<vector<pair<int,int>>> adj(N);
    for (int i = 0; i < N-1; ++i){ // build adj
        adj[from[i]].emplace_back(to[i], weight[i]);
        adj[to[i]].emplace_back(from[i], weight[i]);
    }
    function<array<long long,2>(int,int)> dfs = [&](int cur, int parent){
        array<long long, 2> ans{};
        vector<long long> take, skip, diff;
        for (auto& [next, w] : adj[cur]) if (parent != next){
            auto [not_full, full] = dfs(next, cur);
            take.push_back(not_full + w);
            skip.push_back(full);
            diff.push_back(take.back() - skip.back());
        }
        int n = int(diff.size());
        ranges::nth_element(diff, begin(diff) + k - 1, greater<>());
        ans[0] = reduce(begin(diff), begin(diff) + min(k, n)) + reduce(begin(skip), end(skip));
        if (n && n >= k){
            ans[1] = ans[0];
            ans[0] -= *min_element(begin(diff), begin(diff) + k);
        }
        return ans;
    };
    auto ans = dfs(0, -1);
    return max(ans[0], ans[1]);
}
```

3. Subarray having odd no of divisors

```
#include <iostream>
#include <vector>
#include <unordered_map>
#include <cmath>
using namespace std;
// Function to check if a number is a perfect square
bool isPerfectSquare(long long num) {
    if (num < 0) return false;
    long long root = static_cast<long long>(sqrt(num));
    return root * root == num;
}
int countSubarraysWithOddDivisors(const vector<int>& arr) {
    int n = arr.size();
    unordered_map<long long, int> prefixProductFreq;
    prefixProductFreq[1] = 1; // Initialize the frequency of product 1 to be 1 (for empty subarray)

    long long product = 1;
    int count = 0;
    for (int i = 0; i < n; ++i) {
        product *= arr[i];

        for (auto &entry : prefixProductFreq) {
            long long potentialProduct = product / entry.first;
            if (product % entry.first == 0 && isPerfectSquare(potentialProduct)) {
                count += entry.second;
            }
        }

        prefixProductFreq[product]++;
    }
    return count;
}
int main() {
    vector<int> arr = {1, 2, 3, 4, 5}; // Sample array
    cout << "Total subarrays with product having odd number of divisors: "
         << countSubarraysWithOddDivisors(arr) << endl;
    return 0;
}
```

4. Max Discounts

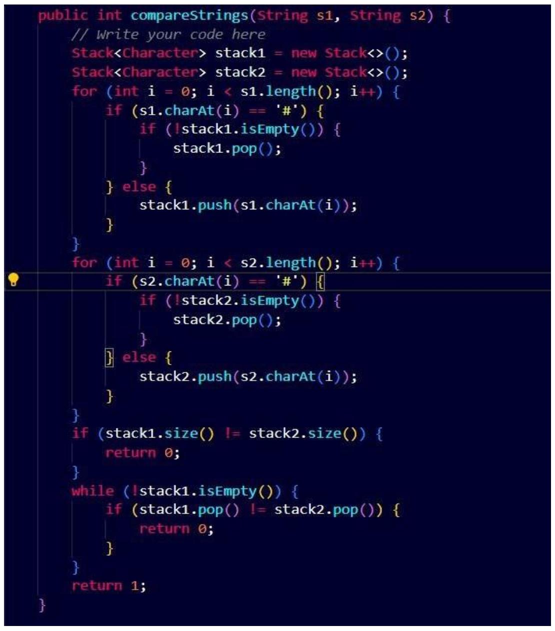
```
#include <iostream>
#include <vector>
using namespace std;
long long calc(int base, int times) {
    return base * (1LL << (times - 1));
}
int main() {
    int n, k;
    cin >> n >> k;
    vector<int> discounts(n);
    for (int i = 0; i < n; ++i) {
        cin >> discounts[i];
    }

    vector<int> times(n, 1);

    for (int i = 0; i < k; ++i) {
        long long max_increase = -1;
        int max_index = -1;
        for (int j = 0; j < n; ++j) {
            long long curr = calc(discounts[j], times[j]);
            long long next = calc(discounts[j], times[j] + 1);
            if (next - curr > max_increase) {
                max_increase = next - curr;
                max_index = j;
            }
        }
        times[max_index]++;
    }

    long long max_discount = 0;
    for (int i = 0; i < n; ++i) {
        max_discount |= calc(discounts[i], times[i]);
    }
    cout << max_discount << endl;
    return 0;
}
```

5. Backspace String Compare



```
public int compareStrings(String s1, String s2) {
    // Write your code here
    Stack<Character> stack1 = new Stack<>();
    Stack<Character> stack2 = new Stack<>();
    for (int i = 0; i < s1.length(); i++) {
        if (s1.charAt(i) == '#') {
            if (!stack1.isEmpty()) {
                stack1.pop();
            }
        } else {
            stack1.push(s1.charAt(i));
        }
    }
    for (int i = 0; i < s2.length(); i++) {
        if (s2.charAt(i) == '#') {
            if (!stack2.isEmpty()) {
                stack2.pop();
            }
        } else {
            stack2.push(s2.charAt(i));
        }
    }
    if (stack1.size() != stack2.size()) {
        return 0;
    }
    while (!stack1.isEmpty()) {
        if (stack1.pop() != stack2.pop()) {
            return 0;
        }
    }
    return 1;
}
```

6. Binary Manipulations

```
def min_operations_to_zero(n):
    bin_str = bin(n)[2:]
    length = len(bin_str)
    operations = 0
    i = 0
    while i < length:
        if bin_str[i] == '1':
            if all(c == '0' for c in bin_str[i + 1:]):
                operations += 1
                bin_str = bin_str[:i] + '0' * (length - i)
            else:
                operations += 1
                i += 1
        else:
            i += 1
    return operations
```

```
#include <iostream>
using namespace std;
class Solution {
public:
   long minimumOneBitOperations(int n) {
       long multiplier = 1;
       long res = 0;
        while (n > 0) {
            res += (n ^ (n - 1)) * multiplier;
            multiplier *= -1;
            n &= (n - 1);
        }
        return abs(res);
    }
};
```

7. Trilogy

```
#include <bits/stdc++.h>
using namespace std;
long long solve(int N, int T) {
    string num = to_string(N);
    vector<long long> candidates;
    auto isDivisibleBy3 = [](const string& s) {
        int sum = 0;
        for (char c : s) sum += (c - '0');
        return sum % 3 == 0;
    };
    for (char d1 = '0'; d1 <= '9'; ++d1) {
        for (char d2 = '0'; d2 <= '9'; ++d2) {
            if (d1 == d2) continue;
            for (size_t i = 0; i <= num.size(); ++i) {
                for (size_t j = i; j <= num.size(); ++j) {
                    string newNum = num;
                    newNum.insert(newNum.begin() + i, d1);
                    newNum.insert(newNum.begin() + j + 1, d2);
                    if (newNum[0] != '0' && isDivisibleBy3(newNum)) {
                        candidates.push_back(stoll(newNum));
                    }
                }
            }
        }
    }
    if (candidates.empty()) return -1;
    if (T == 0) return *min_element(candidates.begin(), candidates.end());
    return *max_element(candidates.begin(), candidates.end());
}
```

```
int main() {
    int N, T;
    cin >> N >> T;
    cout << solve(N, T) << endl;
    return 0;
}
```

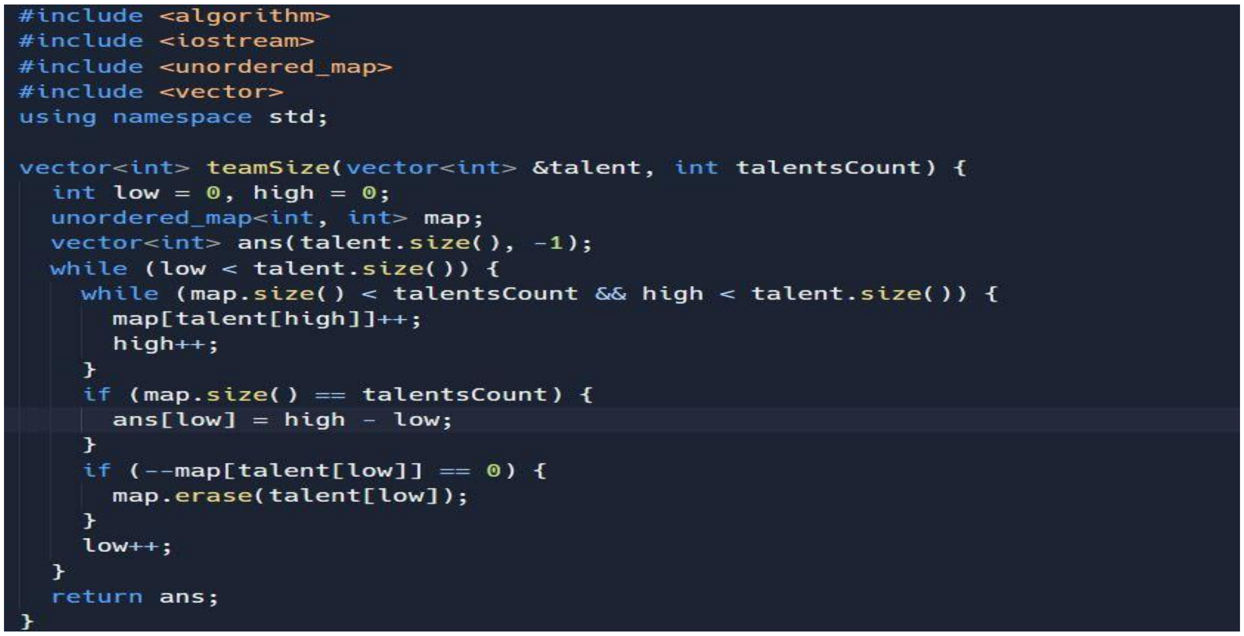
8. Quiz Complettion

```
vector<int> findMinTeamLengths(int tCount, vector<int>& tList) {
    int n = tList.size();
    vector<int> res(n, -1);
    unordered_map<int, int> tMap;
    int dCount = 0;
    int l = 0;
    for (int r = 0; r < n; r++) {
        int curT = tList[r];

        if (tMap[curT] == 0) {
            dCount++;
        }
        tMap[curT]++;

        while (dCount == tCount) {
            res[l] = r - l + 1;
            int lT = tList[l];
            tMap[lT]--;
            if (tMap[lT] == 0) {
                dCount--;
            }
            l++;
        }
    }
    return res;
}
```

//quiz competition



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

vector<int> teamSize(vector<int> &talent, int talentsCount) {
  int low = 0, high = 0;
  unordered_map<int, int> map;
  vector<int> ans(talent.size(), -1);
  while (low < talent.size()) {
    while (map.size() < talentsCount && high < talent.size()) {
      map[talent[high]]++;
      high++;
    }
    if (map.size() == talentsCount) {
      ans[low] = high - low;
    }
    if (--map[talent[low]] == 0) {
      map.erase(talent[low]);
    }
    low++;
  }
  return ans;
}
```

9. Do they Belong

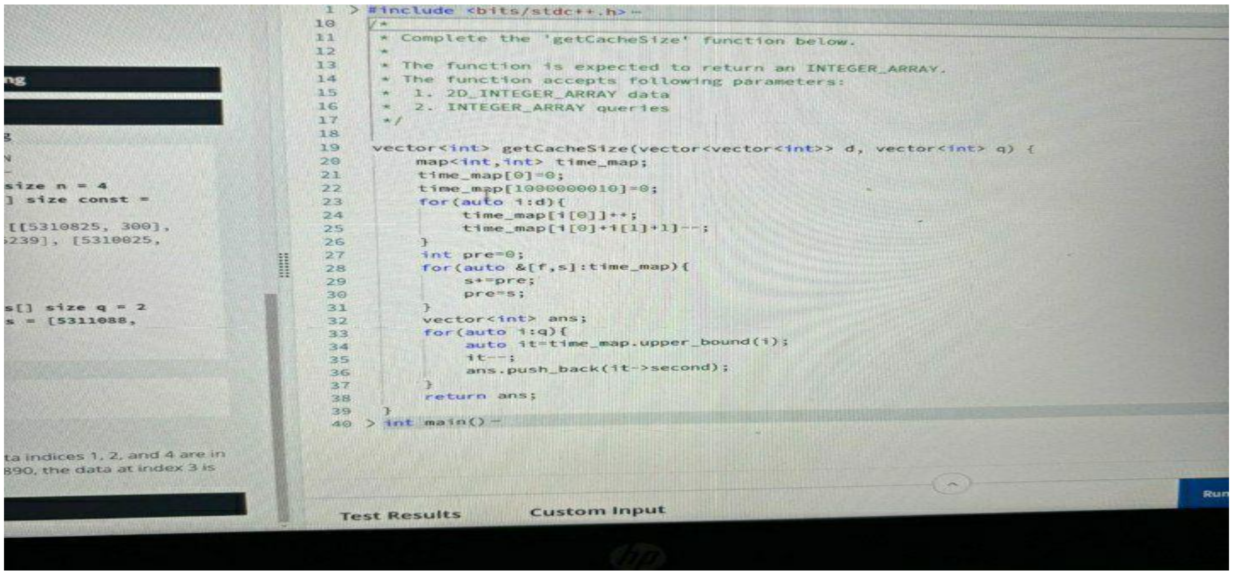
```
import numpy as np
def area(p1, p2, p3):
    x1, y1 = p1
    x2, y2 = p2
    x3, y3 = p3
    return abs((x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2)) / 2.0)
def isInside(a, b, c, p):
    A = area(a, b, c)
    A1 = area(b, c, p)
    A2 = area(a, c, p)
    A3 = area(b, a, p)
    return A == A1 + A2 + A3
a = eval(input("1 = a(x1, y1): "))
b = eval(input("2 = b(x2, y2): "))
c = eval(input("3 = c(x3, y3): "))
p = eval(input("p = p(xp, yp): "))
q = eval(input("q = q(xq, yq): "))
ab = np.linalg.norm([a[0] - b[0], a[1] - b[1]])
bc = np.linalg.norm([c[0] - b[0], c[1] - b[1]])
ac = np.linalg.norm([a[0] - c[0], a[1] - c[1]])
output = 0
if ab + bc > ac and bc + ac > ab and ab + ac > bc:
    if isInside(a, b, c, p) and isInside(a, b, c, q):
        output = 3
    elif isInside(a, b, c, p):
        output = 1
    elif isInside(a, b, c, q):
        output = 2
    else:
        output = 4
print(output)
```

```
public static int pointsBelong(int x1, int y1, int x2, int y2, int x3, int y3, int xp, int yp, int xq, int yq) {
        if (!isValidTriangle(x1, y1, x2, y2, x3, y3)) {
            return 0;
        }
        boolean pBelongs = isPointInTriangle(x1, y1, x2, y2, x3, y3, xp, yp);
        boolean qBelongs = isPointInTriangle(x1, y1, x2, y2, x3, y3, xq, yq);
        if (pBelongs && qBelongs) {
            return 3;
        } else if (pBelongs) {
            return 1;
        } else if (qBelongs) {
            return 2;
        } else {
            return 4;
        }
    }
    private static boolean isValidTriangle(int x1, int y1, int x2, int y2, int x3, int y3) {
        double ab = distance(x1, y1, x2, y2);
        double bc = distance(x2, y2, x3, y3);
        double ac = distance(x1, y1, x3, y3);
        return (ab + bc > ac) && (bc + ac > ab) && (ac + ab > bc);
    }
    private static double distance(int x1, int y1, int x2, int y2) {
        return Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));
    }
    private static boolean isPointInTriangle(int x1, int y1, int x2, int y2, int x3, int y3, int xp, int yp) {
        double areaABC = triangleArea(x1, y1, x2, y2, x3, y3);
        double areaPAB = triangleArea(xp, yp, x1, y1, x2, y2);
        double areaPAC = triangleArea(xp, yp, x1, y1, x3, y3);
        double areaPBC = triangleArea(xp, yp, x2, y2, x3, y3);
        return Math.abs((areaPAB + areaPAC + areaPBC) - areaABC) < 1e-9;
    }
    private static double triangleArea(int x1, int y1, int x2, int y2, int x3, int y3) {
        return Math.abs(x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2)) / 2.0;
```

10. Rearrange Students

```
long long minCost(vector<int>& a, vector<int>& b) {
        map<int,int>mp;
        int n=a.size();
        int mini=INT_MAX;
        for(int i=0;i<n;i++){
            mp[a[i]]++;
            mp[b[i]]--;
            mini=min(mini,a[i]);
            mini=min(mini,b[i]);
        }
        vector<int>x;
        for(auto it:mp){
            int t=it.second;
            if(t%2==1)return -1;
            else{
                for(int i=0;i<abs(t)/2;i++){
                    x.push_back(it.first);
                }
            }
        }
        long long ans=0;
        int m=x.size();
        for(int i=0;i<m/2;i++){
            ans+=min(x[i],2*mini);
        }
        return ans;
    }
//rearrange students
```

11. TTL Cache



```
ng
g
N
size n = 4
] size const =
[[5310825, 300],
5239], [5310825,
s[] size q = 2
s = [5311088,
```

ta indices 1, 2, and 4 are in
890, the data at index 3 is

```
#include <bits/stdc++.h>
/*
 * Complete the 'getCacheSize' function below.
 *
 * The function is expected to return an INTEGER_ARRAY.
 * The function accepts following parameters:
 *  1. 2D_INTEGER_ARRAY data
 *  2. INTEGER_ARRAY queries
 */

vector<int> getCacheSize(vector<vector<int>> d, vector<int> q) {
    map<int,int> time_map;
    time_map[0]=0;
    time_map[1000000010]=0;
    for(auto i:d){
        time_map[i[0]]++;
        time_map[i[0]+i[1]+1]--;
    }
    int pre=0;
    for(auto &[f,s]:time_map){
        s+=pre;
        pre=s;
    }
    vector<int> ans;
    for(auto i:q){
        auto it=time_map.upper_bound(i);
        it--;
        ans.push_back(it->second);
    }
    return ans;
}
int main()
```

12. String Subsequence

```
def count_subsequences(s1, s2):
    m, n = len(s1), len(s2)
    dp = [[0] * (n + 1) for _ in range(m + 1)]
    for j in range(n + 1):
        dp[0][j] = 1
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if s1[i - 1] == s2[j - 1]:
                dp[i][j] = dp[i][j - 1] + dp[i - 1][j - 1]
            else:
                dp[i][j] = dp[i][j - 1]
    return dp[m][n]
//string subsequence
```

13. Get the group

```
#include <iostream>
#include <vector>
#include <string>
#include <unordered_map>
#include <unordered_set>
using namespace std;
void findCircle(int root, unordered_set<int>& seen, unordered_map<int, vector<int>>& graph) {
    for (int friendVertex : graph[root]) {
        if (!seen.count(friendVertex)) {
            seen.insert(friendVertex);
            findCircle(friendVertex, seen, graph);
        }
    }
}
vector<int> getTheGroups(int n, vector<string> queryType, vector<int> student1, vector<int> student2) {
    vector<int> ans;
    unordered_map<int, vector<int>> graph;
    for (int i = 1; i <= n; i++) {
        graph[i] = vector<int>();
    }
    for (int i = 0; i < queryType.size(); i++) {
        string type = queryType[i];
        int f1 = student1[i];
        int f2 = student2[i];
        if (f1 != f2 && type == "Friend") {
            graph[f1].push_back(f2);
            graph[f2].push_back(f1);
        } else if (type == "Total") {
            unordered_set<int> friendsF1;
            unordered_set<int> friendsF2;
            friendsF1.insert(f1);
            friendsF2.insert(f2);
            findCircle(f1, friendsF1, graph);
            findCircle(f2, friendsF2, graph);
            if (f1 == f2) {
                ans.push_back(friendsF1.size());
            } else if (friendsF1.count(f2)) {
                ans.push_back(friendsF1.size());
            } else {
                ans.push_back(friendsF1.size() + friendsF2.size());
            }
        }
    }
    return ans;
```

14. Best Sum Any tree Path

```
max(int a, intb){
if(a>b){
return a;
}
else{
return b;
}
}
long long result = 0;
long long findMaximumPathSum(int currentNode,
                    int previousNode,
                    const vector<vector<int>> &adj,
                    const vector<int> &A)
{

          const vector<int> &v = adj[currentNode];
          int maximumBranchSum1 = 0;
          int maximumBranchSum2 = 0;
          for (auto vtx : v) {

          if (vtx == previousNode) {
          continue;
          }
          long long bs = findMaximumPathSum(vtx, currentNode,
                    adj, A);
           if (bs >= maximumBranchSum1) {
          maximumBranchSum2 = maximumBranchSum1;
          maximumBranchSum1 = bs;
          }
          else {
          maximumBranchSum2
          = max(maximumBranchSum2, bs);
          }
          }
          result = max(result,
          A[currentNode] + maximumBranchSum1
          + maximumBranchSum2);
          return A[currentNode] + maximumBranchSum1;
}
long long bestSumAnyTreePath(vector<int>& parents, vector<int>& values) {
          int n = parents.size();
          if(n == 1){
          return -5;
          }
          vector<vector<int>> mp(n);
          for (int i = 0; i < n; i++) {
          if (parents[i] != -1) {
          mp[parents[i]].push_back(i);
          }
          }
          result = 0;
          findMaximumSumPath(0, -1, mp, values);
          return result;
}

#include <vector>
#include <algorithm>
#include <climits>
using namespace std;
int bestSumAnyTreePath(vector<int> parent, vector<int> values) {
          int n = parent.size();
          vector<int> dp(n, INT_MIN);
          int max_sum = INT_MIN;
          for (int i = 0; i < n; i++) {
          if (parent[i] != -1) {
          dp[i] = max(dp[i], values[i] + max(dp[parent[i]], 0));
          } else {
          dp[i] = values[i];
          }
```

```
        max_sum = max(max_sum, dp[i]);
        }
        return max_sum;
}
```

15. Whole minute Dillema

```
#include <iostream>
#include <vector>
#include <unordered_map>
using namespace std;
int playlist(int n, vector<int> songs) {
        vector<int> remainder_count(60, 0);
        for (int song : songs) {
        int remainder = song % 60;
        remainder_count[remainder]++;
        }
        int pairs = 0;
        pairs += (remainder_count[0] * (remainder_count[0] - 1)) / 2;
        pairs += (remainder_count[30] * (remainder_count[30] - 1)) / 2;
        for (int i = 1; i <= 29; i++) {
        pairs += remainder_count[i] * remainder_count[60 - i];
        }
        return pairs;
}
```

16. Modest Number

```
#include <iostream>
#include <vector>
bool isModest(int num) {
        for (int i = 10; i <= 100000; i*= 10) {
        int left = num%i;
        int right = num/i;
        if (left == 0) continue;
        if (right == 0) break;
        if (num % left == right) return true;
        }
        return false;
}
int main() {
        int M, N;
        std::cin >> M >> N;
        std::vector<int> modestNumbers;
        for (int num = M; num <= N; num++) {
        if (isModest(num)) {
        modestNumbers.push_back(num);
        }
        }
        if (modestNumbers.empty()) {
        std::cout << "No modest numbers found within the range";
        } else {
        for (size_t i = 0; i < modestNumbers.size(); i++) {
        std::cout << modestNumbers[i];
        if (i < modestNumbers.size() - 1) {
        std::cout << " ";
        }
        }
        }
        return 0;
}
```

17. Valid BST Permutations

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const ll mod = 1e8+7;
ll multmod(ll a, ll b) {
    ll result = 0;
    while (b) {
        if (b & 1)
            result = (result + a) % mod;
        a = (a + a) % mod;
        b >>= 1;
    }
    return result;
}
ll addmod(ll a, ll b) {
    a = a % mod;
    b = b % mod;
    return (((a + b) % mod) + mod) % mod;
}
vector<int> numBST(vector<int> nodeValues) {
    ll maxVal = *max_element(nodeValues.begin(), nodeValues.end());
    vector<unsigned ll> dp(maxVal + 1);
    dp[0] = 1;
    dp[1] = 1;
    for (ll i = 2; i <= maxVal; i++) {
        unsigned ll res = 0;
        for (ll j = 0; j < i; j++)
            res = addmod(res, multmod(dp[j], dp[i - j - 1]));
        dp[i] = res % mod;
    }
    vector<int> ans(nodeValues.size());
    for (ll i = 0; i < nodeValues.size(); i++) {
        ans[i] = dp[nodeValues[i]] % mod;
    }
    return ans;
}
int main() {
    ll n;
    cin >> n;
    vector<int> nodeValues(n);
    for (ll i = 0; i < n; i++) cin >> nodeValues[i];
    vector<int> ans = numBST(nodeValues);
    for (auto it : ans) cout << it << " ";
    return 0;
}
```

18. 18.