Text

Description automatically generated with medium confidence

Text

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

#include<bits/stdc++.h>

#define ll long long int

#define maxm 1e6

#define f(i, in, n) for(long long int i=in; i<n; i++)

#define fl(i, in, n) for(long long int i=n-1; i>=in; i--)

#define lim(i, a, b) for (long long int i=a; i<b; i++)

using namespace std;

vector<ll> xor\_of(300001);

vector<ll> seg\_tree(maxm);

// Building a Segment Tree

ll buildSegmentTree(vector<ll> v, ll i, ll l, ll r) {

    if (l==r) {

        seg\_tree[i]=v[l];

        return v[l];

    }

    ll mid=(l+r)/2;

    seg\_tree[i]=buildSegmentTree(v, 2\*i+1, l, mid)+buildSegmentTree(v, 2\*i+2, mid+1, r);

    return seg\_tree[i];

}

// To get sum from Range Query

ll getSum(ll index, ll st, ll en, ll l, ll r) {

    if (st>r or en<l) return 0;

    if (l<=st and en<=r) return seg\_tree[index];

    ll mid=(st+en)/2;

    return getSum(2\*index+1, st, mid, l, r)+getSum(2\*index+2, mid+1, en, l, r);

}

// Update Queries

void updateRange(ll index, ll pos, ll diff, ll st, ll en) {

    if (st>pos or en<pos) return;

    seg\_tree[index]+=diff;

    if (st!=en) {

        ll mid=(st+en)/2;

        updateRange(2\*index+1, pos, diff, st, mid);

        updateRange(2\*index+2, pos, diff, mid+1, en);

    }

}

int power(long long x, unsigned int y, int p) {

    int res = 1;     // Initialize result

    x = x % p; // Update x if it is more than or

                // equal to p

    if (x == 0) return 0; // In case x is divisible by p;

    while (y > 0)

    {

        // If y is odd, multiply x with result

        if (y & 1)

            res = (res\*x) % p;

        // y must be even now

        y = y>>1; // y = y/2

        x = (x\*x) % p;

    }

    return res;

}

class Point {

public:

    int x, y;

    Point(int a=0, int b=0) : x(a),y(b) {}

};

double distance\_bw\_points(Point P1, Point P2) {

    // Calculating distance

    double x1=P1.x, y1=P1.y;

    double x2=P2.x, y2=P2.y;

    return sqrt(pow(x2 - x1, 2) +

                pow(y2 - y1, 2) \* 1.0);

}

Point mirrorImage(double a, double b, double c, Point X) {

    double temp = -2 \* (a \* X.x + b \* X.y + c) /

                              (a \* a + b \* b);

    double p = temp \* a + X.x;

    double q = temp \* b + X.y;

    Point P(p, q);

    return P;

}

Point find\_Centroid(vector<Point> &v) {

    pair<double, double> ans = { 0, 0 };

    int n = v.size();

    double signedArea = 0;

    // For all vertices

    for (int i = 0; i < v.size(); i++) {

        double x0 = v[i].x, y0 = v[i].y;

        double x1 = v[(i + 1) % n].x, y1 =

                            v[(i + 1) % n].y;

        // Calculate value of A

        // using shoelace formula

        double A = (x0 \* y1) - (x1 \* y0);

        signedArea += A;

        // Calculating coordinates of

        // centroid of polygon

        ans.first += (x0 + x1) \* A;

        ans.second += (y0 + y1) \* A;

    }

    signedArea \*= 0.5;

    ans.first = (ans.first) / (6 \* signedArea);

    ans.second = (ans.second) / (6 \* signedArea);

    Point P(ans.first, ans.second);

    return P;

}

//utility function to find GCD of two numbers

// GCD of a and b

int gcd(int a, int b)

{

    if (b == 0)

       return a;

    return gcd(b, a%b);

}

// Finds the no. of Integral points between

// two given points.

int getBoundaryCount(Point p,Point q)

{

    // Check if line parallel to axes

    if (p.x==q.x)

        return abs(p.y - q.y) - 1;

    if (p.y == q.y)

        return abs(p.x - q.x) - 1;

    return gcd(abs(p.x-q.x), abs(p.y-q.y)) - 1;

}

// Returns count of points inside the triangle

int getInternalCount(Point &p, Point &q, Point &r)

{

    // 3 extra integer points for the vertices

    int BoundaryPoints = getBoundaryCount(p, q) +

                         getBoundaryCount(p, r) +

                         getBoundaryCount(q, r) + 3;

    // Calculate 2\*A for the triangle

    int doubleArea = abs(p.x\*(q.y - r.y) + q.x\*(r.y - p.y)  +

                         r.x\*(p.y - q.y));

    // Use Pick's theorem to calculate the no. of Interior points

    return (doubleArea - BoundaryPoints + 2)/2;

}

vector<long long> nextSmallerElement(long long arr[], int n) {

        // Your code here

    stack<long long> s;

    vector<long long> ans1(n);

    for (int i=n-1; i>=0; i--) {

        while (!s.empty() and arr[i]<=arr[s.top()]) {

            s.pop();

        }

        if (s.empty()) ans1[i]=n;

        else ans1[i]=s.top();

        s.push(i);

    }

    return ans1;

}

vector<long long> nextSmallerElementfromBegin(long long arr[], int n) {

    vector<long long> ans2(n);

    stack<long long> s1;

    for (int i=0; i<n; i++) {

        while (!s1.empty() and arr[i]<=arr[s1.top()]) {

            s1.pop();

        }

        if (s1.empty()) ans2[i]=0;

        else ans2[i]=s1.top();

        s1.push(i);

    }

    return ans2;

}

    long long getMaxArea(long long arr[], int n) {

        // Your code here

        vector<long long> aux1=nextSmallerElement(arr, n);

        vector<long long> aux2=nextSmallerElementfromBegin(arr, n);

        long long ans=0, area;

        for (int i=0; i<n-1; i++) {

            if (arr[i]>arr[i+1]) {

                long long width=aux1[i+1]-(i+1);

                area=arr[i+1]\*(width+1);

                ans=max(ans, area);

            }

            else {

                long long width=aux1[i]-i;

                area=arr[i]\*width;

                ans=max(ans, area);

            }

        }

        ans=max(ans, arr[n-1]);

        ans=max(ans, arr[0]);

        for (int i=1; i<n; i++) {

            if (arr[i]<arr[i-1]) {

                long long width=i-aux2[i]+1;

                area=arr[i]\*width;

                ans=max(ans, area);

            }

            else {

                long long width=i-aux2[i-1];

                area=arr[i-1]\*(width+1);

                ans=max(ans, area);

            }

        }

        return ans;

    }

/\*string decToBinary(int n) {

    // Size of an integer is assumed to be 32 bits

    string ans="";

    for (int i = 31; i >= 0; i--) {

        int k = n >> i;

        if (k & 1)

            ans+='1';

        else

            ans+='0';

    }

    return ans;

}\*/

ll minCost(vector<ll> v, ll k, ll x) {

    sort(v.begin(), v.end());

    ll ans=0;

    while (v.size()>1) {

        ll n=v.size();

        if (k>0 and x<=v[n-1]+v[n-2]) {

            ans+=x;

            v.pop\_back();

            v.pop\_back();

            k--;

        }

        else {

            ans+=(v[n-1]+v[n-2]);

            v.pop\_back();

            v.pop\_back();

        }

    }

    if (v.size()==1) ans+=v[v.size()-1];

    return ans;

}

ll fun(string s, ll k) {

    unordered\_map<char, ll> m;

    f(i, 0, s.length()) {

        int x=s[i]-'0';

        if (x==1 or (x!=2 and x%2==0) or x==9) return (ll)x;

        m[s[i]]++;

    }

    for (auto it : m) {

        if (it.second>1) {

            ll u=it.first-'0';

            return u+(10\*u);

        }

    }

    f(i, 0, s.length()) {

        f(j, i+1, s.length()) {

            ll y=10\*(s[i]-'0')+(s[j]-'0');

            if (y%2==0 or y%3==0 or y%5==0 or y%7==0) return y;

        }

    }

}

unsigned ll num(ll s, ll x) {

    ll z=(s-x)/2;

    ll ct=0;

    if (z<0 or (s-x)%2!=0) return 0;

    for (ll i=31; i>=0; i--) {

        if ((x&(1<<i))!=0 and (z&(1<<i))!=0) return 0;

        if ((x&(1<<i))!=0 and (z&(1<<i))==0) ct++;

    }

    if (z==0) ct--;

    ll ans=(1<<ct);

    return ans;

}

void seiv() {

    xor\_of[0]=0;

    f(i, 1, xor\_of.size()) {

        xor\_of[i]=xor\_of[i-1]^i;

    }

}

bool primeFactors(ll n, ll b) {

    // Print the number of 2s that divide

    if (n % 2 == 0) {

        if (b % 2) return false;

        while (n % 2 == 0) {

            n = n/2;

        }

    }

    // n must be odd at this point. So we can skip

    // one element (Note i = i +2)

    for (ll i = 3; i\*i <= n; i = i + 2) {

        // While i divides n, print i and divide n

        if (n % i == 0) {

            if (b % i) return false;

            while (n % i == 0) {

                n = n/i;

            }

        }

    }

    // This condition is to handle the case when n

    // is a prime number greater than 2

    if (n > 2) {

        if (b % n) return false;

    }

    return true;

}

vector<vector<int> > fourSum(vector<int> &arr, int k) {

    // Your code goes here

    vector<vector<int>> vec;

    int n=arr.size();

    sort(arr.begin(), arr.end());

    for (int i=0; i<n-3; i++) {

        for (int j=i+1; j<n-2; j++) {

            int l=i+1, r=n-1;

            int required=k-(arr[i]+arr[j]);

            while (l<r) {

                if (r==j) {

                    r--;

                    continue;

                }

                if (l==j) {

                    l++;

                    continue;

                }

                if (arr[l]+arr[r]==required) {

                    vec.push\_back({arr[i], arr[j], arr[l],  arr[r]});

                    sort(vec.back().begin(), vec.back().end());

                }

                else if (arr[l]+arr[r]>required) r--;

                else l++;

            }

        }

    }

    sort(vec.begin(), vec.end());

    return vec;

}

ll total\_score(vector<ll> a, vector<ll> b) {

    ll n=a.size();

    vector<ll> img(n);

    f(i, 0, n) {

        img[i]=-a[i]\*(i+1);

    }

    f(i, 0, n) {

        img[i]+=b[i]\*(i+1);

    }

    ll score=accumulate(img.begin(), img.end(), 0);

    return score;

}

vector<ll> countGreater(vector<ll> v) {

    multiset<ll> st;

    vector<ll> ans(v.size());

    f(i, 0, v.size()) {

        st.insert(v[i]);

        auto it=upper\_bound(st.begin(), st.end(), v[i]);

        ans[i]=distance(it, st.end());

    }

    return ans;

}

vector<ll> countSmaller(vector<ll> v) {

    multiset<ll> st;

    vector<ll> ans(v.size());

    f(i, 0, v.size()) {

        st.insert(v[i]);

        auto it=lower\_bound(st.begin(), st.end(), v[i]);

        ans[i]=distance(st.begin(), it);

    }

    return ans;

}

bool check(string s) {

    unordered\_map<char, ll> m;

    f(i, 0, s.length()) {

        m[s[i]]++;

    }

    ll ct1=0, ct2=0;

    for (auto it : m) {

        if (it.second==1) ct1++;

        if (it.second%2==0) ct2++;

    }

    return ct2>=ct1;

}

int main() {

    ios\_base::sync\_with\_stdio(false);

    cin.tie(NULL);

    int q;

    cin>>q;

    while (q--) {

        ll n, x;

        cin>>n>>x;

        vector<ll> a(n);

        f(i, 0, n) cin>>a[i];

        vector<ll> vf=a;

        sort(vf.begin(), vf.end());

        if (x>n) cout<<"no"<<endl;

        else {

            bool flag=false;

            lim(i, n-x, x) {

                if (a[i]!=vf[i]) {

                    flag=true;

                    break;

                }

            }

            if (flag) cout<<"no"<<endl;

            else cout<<"yes"<<endl;

        }

    }

    return 0;

}