Problem

Vanity has N trinkets on her shelf, numbered 1, 2, ..., N from left to right. Trinkets come in different types, which ar denoted by positive integers. The i-th trinket on her shelf is of type Ai.

She is going to see her family overseas today and would like to bring as many trinkets as she can. However, since sh is in a hurry, Vanity must take a consecutive interval of trinkets. Formally, Vanity selects two indices, l and r, and ta es all of the trinkets numbered l, l+1, ..., r-1, r. Also, due to tax rules, airport security will throw away all trinkets of type if Vanity has more than S of that type in the chosen interval.

For example, suppose that S = 2, and Vanity brings six trinkets: one of type 0, two of type 1, and three of type 2. She will be allowed to keep the trinket of type 0 and both trinkets of type 1, but she will lose all of the trinkets of type 2!

Vanity needs to choose l and r such that she can take the maximum number of trinkets for her family. What is the m ximum number of trinkets she can bring?

Input

The first line of the input gives the number of test cases, T. T test cases follow. The first line of each test case contai s the two integers N and S, the number of trinkets, and the maximum number of trinkets allowed of a single type, res ectively. The second line contains N integers. The i-th integer gives Ai, the type of the i-th trinket.

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is th maximum number of trinkets that Vanity can bring to her family.

Limits

Time limit: 30 seconds per test set.
Memory limit: 1GB.
$1 \leq T \leq 100$.
$1 \leq Ai \leq 105$.
$1 \leq S \leq N$.

Test set 1 (Visible)

$1 \leq N \leq 1000$.

Test set 2 (Hidden)

$1 \leq N \leq 105$.

Sample

Input

Output

```
4
6 2
1 1 4 1 4 4
8 1
1 2 500 3 4 500 6 7
10 1
100 200 8 8 8 8 8 300 400 100
12 2
40 50 1 1 1 60 70 2 2 2 80 90
```

Case #1: 4
Case #2: 6
Case #3: 4
Case #4: 6

In Sample Case #1, Vanity should choose l = 2 and r = 5. This allows her to take 4 trinkets to the airport of types 1, , 1 and 4. None of them are thrown away by airport security, so she is able to bring 4 trinkets to her family.

In Sample Case #2, Vanity should choose l = 1 and r = 8. This allows her to take all 8 trinkets to the airport. Her trin ets of type 500 are thrown away since she has more than S = 1 of them, so she is able to bring a total of 6 trinkets to er family.

In Sample Case #3, Vanity should choose l = 1 and r = 9. This allows her to take 9 trinkets to the airport of types 10 , 200, 8, 8, 8, 8, 8, 300 and 400. Her trinkets of type 8 are thrown away since she has more than S = 1 of them, so sh is able to bring a total of 4 trinkets to her family.

In Sample Case #4, Vanity should choose l = 1 and r = 12. This allows her to take all 12 trinkets to the airport. Her tr nkets of type 1 and 2 are thrown away since she has more than S = 2 of each of them, so she is able to bring a total o 6 trinkets to her family.

Note: We do not recommend using interpreted/slower languages for this problem.

// In the name of God

```
#include <iostream>
#include <algorithm>
#include <fstream>
#include <vector>
#include <deque>
#include <assert.h>
#include <queue>
#include <stack>
#include <set>
#include <map>
#include <stdio.h>
#include <string.h>
#include <utility>
#include <math.h>
#include <bitset>
#include <iomanip>
#include <complex>

using namespace std;

#define rep(i, a, b) for (int i = (a), i##_end_ = (b); i < i##_end_; ++i)
#define debug(...) fprintf(stderr, __VA_ARGS__)
#define mp make_pair
#define x first
#define y second
#define pb push_back
```

```
#define SZ(x) (int((x).size()))
#define ALL(x) (x).begin(), (x).end()

template<typename T> inline bool chkmin(T &a, const T &b) { return a > b ? a = b, 1 : 0; }
template<typename T> inline bool chkmax(T &a, const T &b) { return a < b ? a = b, 1 : 0; }
template<typename T> inline bool smin(T &a, const T &b)   { return a > b ? a = b : a;    }
template<typename T> inline bool smax(T &a, const T &b)   { return a < b ? a = b : a;    }

typedef long long LL;

const int N = (int) 2e5 + 6, mod = (int) 0;
int seg[N << 2], ch[N << 2];
void update(int i, int j, int x, int v = 1, int b = 0, int e = N) {
 if (i >= e || b >= j) return;
 if (i <= b && e <= j) {
  seg[v] += x;
  ch[v] += x;
  return;
 }
 int m = b + e >> 1, l = v << 1, r = l | 1;
 update(i, j, x, l, b, m);
 update(i, j, x, r, m, e);
 seg[v] = max(seg[l], seg[r]) + ch[v];
}
int a[N];
vector<int> all[N];
int main() {
 int tc;
 cin >> tc;
 for (int tt = 1; tt <= tc; ++tt) {
  memset(seg, 0, sizeof seg);
  memset(ch, 0, sizeof ch);
  cout << "Case #" << tt << ": ";
  for (int j = 0; j < N; ++j) all[j].clear();
  int n, s;
  cin >> n >> s;
  int res = 0;
  for (int j = 0; j < n; ++j) cin >> a[j], --a[j], all[a[j]].push_back(j);
  for (int j = 0; j < n; ++j) {
   int cur = a[j];
   update(0, j + 1, 1);
   int pos = lower_bound(ALL(all[cur]), j) - all[cur].begin();
   if (pos >= s) {
    int xl = all[cur][pos - s];
    update(0, xl + 1, -1);
    int bef = (pos - s == 0 ? -1 : all[cur][pos - s - 1]);
    update(bef + 1, xl + 1, -s);
   }
   res = max(res, seg[1]);
  }
  cout << res << '\n';
 }
```

}