```
//Note that this solution was officially accepted!
#include <bits/stdc++.h>
#include <stdio.h>
#include <set>

using namespace std;

//#define N 100003

//Union find specific
int link[100003];
int size[100003];
int answers[101];

int find(int a){
        int node = a;
        while(link[node] != node){
                node = link[node];
                if(node == -1) break;
        }
        return node;
}

int merge(int a, int b){
        int repa = find(a);
        int repb = find(b);

        if(repa == repb) return -1;
        if(size[repa] > size[repb]){
                link[repb] = link[repa];
                size[repa] += size[repb];
                //s.erase(repb);
                return repb;
        } else{
                link[repa] = link[repb];
                size[repb] += size[repa];
                //s.erase(repa);
                return repa;
        }
}

int main(void){
        //Preliminaries
        int T;
        scanf("%d", &T);

        //For each possible level
        for(int t = 0; t<T; t++){
        int answer = -1;

        //Create a set of the possible levels
        set <int> levels;

        //Read in the heights
        int N, X;
        scanf("%d %d", &N, &X);

        //Reset link and size
        for(int i = 1; i<= N; i++){
                link[i] = -1;
                size[i] = 0;
        }

        vector <tuple<int,int>> heights;
        //Read in teh heights
        for(int i = 1; i<=N; i++){
                int height;
                scanf("%d", &height);
                heights.push_back({height, i});
                levels.insert(height);
        }

        //Sort the vector of heights
        sort(heights.begin(), heights.end());
        /*for(int i = 0; i<N; i++){
                printf("%d ", get<0>(heights[i]));
        }*/

        //Create a set which will contain the union heads.
        set <int> unions;

        //Make the 0th and Nth heights black (corresponding to the
edges of the island)
        link[0] = 0;
        size[0] = 1;
        link[N+1] = N+1;
        size[N+1] = 1;
        unions.insert(0);
        unions.insert(N+1);

        //Go through all the possible levels
        int pointer = 0;
        int no_islands = unions.size()-1;
        if(no_islands == X){
                answers[t] = 0;
                continue;
        }
        for(auto level : levels){
                while(get<0>(heights[pointer]) == level){
                        int position = get<1>(heights[pointer]);
                        link[position] = position;
                        size[position] = 1;
                        unions.insert(position);
                        int rep_del;
                        if(find(position-1) != -1) {rep_del =
merge(position, position-1); unions.erase(rep_del);}
                        if(find(position+1) != -1) {rep_del =
merge(position, position+1); unions.erase(rep_del);}
```

```
                        pointer++;
                        if(pointer >= N) break;
                }
                int no_islands = unions.size()-1;
                if(no_islands == X){
                        answer = level;
                        break;
                }
        }

        answers[t] = answer;
        }

        for(int i = 0; i<T; i++){
                printf("%d\n", answers[i]);
        }

        return 0;
}
```