Problem
You may have heard of Google's Tensor Processing Units, which are used to build neural networks. However, there s one research area that is even deeper and more important than machine learning: sorting!

We are working on a special new chip called the Sorting Permutation Unit, which is very fast at applying permutatio s to arrays of integers. Formally, a permutation is an ordering of the first n positive integers

p1, p2, ..., pn

and applying it to an array of n integers

a1, a2, ..., an

yields the new array

ap1, ap2, ..., apn.

For example, applying the permutation 3, 1, 2, 4 to the array 99, 234, 45, 800 would yield the new array 45, 99, 234, 800.

However, permutations are expensive to represent in the hardware, so the unit can only have access to at most P disti ct permutations. We need your help figuring out what those permutations should be!

Given K arrays of N integers each, you must first specify up to P permutations (of size N) of your choice. Then, for ach of those K input arrays, you must provide one sequence of up to S instructions (each of which is a permutation f om your specified set). When the instructions in this sequence are applied, in the given order, to the array, they must ield an array sorted in nondecreasing order. In each of your K sequences of instructions, you may use each of your P permutations zero or more times (not necessarily consecutively).

Input
The first line of the input gives the number of test cases, T. T test cases follow. Each begins with one line with four i tegers P, S, K, and N: the maximum number of permutations allowed, the maximum number of instructions you are llowed to use to sort each array, the number of arrays, and the number of integers in each array. Then, there are K m re lines of N integers Ai1, Ai2, .., AiN each, where the j-th integer on the i-th line, Aij, represents the j-th value of th i-th array.

Output
For each test case, first output the following, in this order:

One line containing Case #x:, where x is the test case number (starting from 1).
One line containing one integer P', where 1 ≤ P' ≤ P: the number of permutations you have chosen to use.
P' lines, the i-th of which contains N integers pi1 pi2 ... piN, where pij is the j-th element of the i-th permutation.
Then, output K more lines. The i-th of these gives the instructions that you will apply to the i-th array given in the in ut. Each such line must begin with one integer S', where 0 ≤ S' ≤ S, and must continue with S' integers X1, X2, ..., X ', where 1 ≤ Xk ≤ P' for all k. Here, Xk represents that the k-th instruction you apply to the i-th array is the Xk-th per utation (counting starting from 1) in your list of permutations. These instructions must yield an array with the eleme ts of the i-th input array, sorted in nondecreasing order.

Limits
Time limit: 20 seconds per test set.
Memory limit: 1GB.
1 ≤ T ≤ 10.

S = 450.
1 ≤ K ≤ 30.
2 ≤ N ≤ 50.
1 ≤ Aij ≤ 1000, for all i and j.

Test set 1 (Visible)
P = 20.

Test set 2 (Hidden)
P = 5.

Sample

Input

Output

```
2
20 450 4 3
10 10 11
17 4 1000
999 998 997
10 10 11
20 450 5 5
1 2 3 4 5
2 3 4 5 1
3 4 5 1 2
4 5 1 2 3
5 1 2 3 4
```

```
Case #1:
2
3 1 2
2 1 3
0
1 2
2 2 1
1 2
Case #2:
1
5 1 2 3 4
0
1 1
2 1 1
3 1 1 1
4 1 1 1 1
```

In Sample Case #1, we can define up to P = 20 permutations. One viable strategy uses only these two:

3 1 2

2 1 3
We can handle the four arrays as follows:

10 10 11: This is already sorted in nondecreasing order, so we do not need to do anything.
17 4 1000: We can apply permutation #2 to yield 4 17 1000.
999 998 997: One option is to first apply permutation #2 to turn the array into 998 999 997, then apply permutation 1 to turn the array into 997 998 999.
10 10 11: This is the same as the first array. Applying permutation #2 also yields array sorted in nondecreasing order (But we could have used the line 0 as we did before.)
In Sample Case #2, notice that we can use the same permutation instruction more than once on the same array, if des red.

Solution:

```
#include <bits/stdc++.h>

using namespace std;

template <typename A, typename B>
string to_string(pair<A, B> p);

template <typename A, typename B, typename C>
string to_string(tuple<A, B, C> p);

template <typename A, typename B, typename C, typename D>
string to_string(tuple<A, B, C, D> p);

string to_string(const string& s) {
  return '"' + s + '"';
}

string to_string(const char* s) {
  return to_string((string) s);
}

string to_string(bool b) {
  return (b ? "true" : "false");
}

string to_string(vector<bool> v) {
  bool first = true;
  string res = "{";
  for (int i = 0; i < static_cast<int>(v.size()); i++) {
    if (!first) {
      res += ", ";
    }
    first = false;
    res += to_string(v[i]);
  }
  res += "}";
  return res;
}

template <size_t N>
string to_string(bitset<N> v) {
  string res = "";
  for (size_t i = 0; i < N; i++) {
    res += static_cast<char>('0' + v[i]);
  }
  return res;
}

template <typename A>
string to_string(A v) {
  bool first = true;
  string res = "{";
  for (const auto &x : v) {
    if (!first) {
      res += ", ";
    }
    first = false;
    res += to_string(x);
  }
  res += "}";
  return res;
}

template <typename A, typename B>
string to_string(pair<A, B> p) {
  return "(" + to_string(p.first) + ", " + to_string(p.second) + ")";
}

template <typename A, typename B, typename C>
string to_string(tuple<A, B, C> p) {
  return "(" + to_string(get<0>(p)) + ", " + to_string(get<1>(p)) + ", " + to_string(get<2>(p)) + ")";
}

template <typename A, typename B, typename C, typename D>
string to_string(tuple<A, B, C, D> p) {
  return "(" + to_string(get<0>(p)) + ", " + to_string(get<1>(p)) + ", " + to_string(get<2>(p)) + ", " + to_string(get<3
(p)) + ")";
}

void debug_out() { cerr << endl; }

template <typename Head, typename... Tail>
void debug_out(Head H, Tail... T) {
  cerr << " " << to_string(H);
  debug_out(T...);
}

#ifdef LOCAL
#define debug(...) cerr << "[" << #__VA_ARGS__ << "]:", debug_out(__VA_ARGS__)
```

```
#else
#define debug(...) 42
#endif

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout << fixed << setprecision(17);
  int tt;
  cin >> tt;
  for (int qq = 1; qq <= tt; qq++) {
    cout << "Case #" << qq << ":" << '\n';
    int p, s, k, n;
    cin >> p >> s >> k >> n;
    vector<vector<int>> tests(k, vector<int>(n));
    for (int i = 0; i < k; i++) {
      vector<pair<int, int>> a(n);
      for (int j = 0; j < n; j++) {
        cin >> a[j].first;
        a[j].second = j;
        tests[i][j] = a[j].first;
      }
      sort(a.begin(), a.end());
      for (int j = 0; j < n; j++) {
//        tests[i][a[j].second] = j;
      }
    }
    vector<int> shifts = {1, min(3, n), min(9, n)};
    vector<int> takes = {1, min(5, n - 1)};
    vector<vector<int>> perm(shifts.size() + takes.size(), vector<int>(n));
    for (int i = 0; i < (int) shifts.size(); i++) {
      for (int j = 0; j < n; j++) {
        perm[i][j] = (j + shifts[i]) % n;
      }
    }
    for (int i = 0; i < (int) takes.size(); i++) {
      int id = (int) shifts.size() + i;
      perm[id][0] = 0;
      for (int j = 1; j < n; j++) {
        perm[id][j] = 1 + ((j - 1 + n - takes[i] - 1) % (n - 1));
      }
    }
    cout << (int) perm.size() << '\n';
    for (int i = 0; i < (int) perm.size(); i++) {
      for (int j = 0; j < n; j++) {
        if (j > 0) {
          cout << " ";
        }
        cout << perm[i][j] + 1;
      }
      cout << '\n';
    }
    for (auto vvv : tests) {
      bool found = false;
      for (int init = 0; init < n; init++) {
        auto v = vvv;
        vector<int> seq;
        auto Apply = [&](int id) {
          seq.push_back(id);
          vector<int> new_v(n);
          for (int i = 0; i < n; i++) {
            new_v[i] = v[perm[id][i]];
          }
          swap(v, new_v);
        };
        auto ManyShifts = [&](int cnt) {
//          debug("shifts", cnt);
          for (int i = (int) shifts.size() - 1; i >= 0; i--) {
            while (cnt >= shifts[i]) {
              Apply(i);
              cnt -= shifts[i];
            }
          }
        };
        auto ManyTakes = [&](int cnt) {
//          debug("takes", cnt);
          for (int i = (int) takes.size() - 1; i >= 0; i--) {
            while (cnt >= takes[i]) {
              Apply((int) shifts.size() + i);
              cnt -= takes[i];
            }
          }
        };
        ManyShifts(init);
        for (int i = 0; i < n; i++) {
          int ptr = n - 1;
          int steps = 0;
          for (int j = 0; j < i; j++) {
            if (v[n - 1 - j] > v[0]) {
              --ptr;
              ++steps;
            } else {
              break;
            }
          }
          ManyTakes(steps);
          ManyShifts(steps + 1);
        }
        for (int i = 0; i < n - 1; i++) {
          assert(v[i] <= v[i + 1]);
        }
        if ((int) seq.size() <= s) {
          cout << seq.size();
          for (int x : seq) {
```

```
          cout << " " << x + 1;
        }
        cout << '\n';
        found = true;
        break;
      }
    }
    assert(found);
  }
 }
 return 0;
}
```