```cpp
#include <iostream>
#include <cmath>
#include <vector>
#include <algorithm>

// By this point in the course standard classes weren't introduced.
// Nevertheless it is simpler to manage an array of arbitrary size by using
// the class std::vector, which manages the memory allocation internally.

// It also makes it easier to get the maximum element.

int cont(std::vector<unsigned> v, int m, int M, int n);
void compute_intervals(const std::vector<unsigned>, unsigned, unsigned, int);

int main() {
    unsigned l, n;

    std::cout << "Number of intervals: ";
    std::cin >> l;

    std::cout << "Size of dataset: ";
    std::cin >> n;

    std::cout << "Input data: ";

    std::vector<unsigned> v;
    unsigned x;
    for (auto t = 0; t < n; t++) {
        std::cin >> x;
        v.push_back(x);
    }

    int MAX = *max_element(std::begin(v), std::end(v));

    // COMPUTE INTERVALS
    compute_intervals(v, n, l, MAX);

    return 0;
}


int cont(std::vector<unsigned> v, int m, int M, int n) {
    int tot = 0;
    for (auto n : v)
        if (n >= m && n < M)
            tot++;
    return tot;
}


void compute_intervals(const std::vector<unsigned> v, unsigned n, unsigned l, int M) {
    int k, i;
    k = ceil((double)M / l);
    for (i = 0; i < (l - 1); i++)
        std::cout << i * k << " : " << cont(v, (i * k), ((i + 1) * k), n) << "\n";
    std::cout << i * k << " : " << cont(v, (i * k), (M + 1), n) << "\n";
}
```