

```

1  #include <iostream>
2  #include <cmath>
3  #include <algorithm>
4
5  int dumbMax(unsigned [], int);
6  int smartMax(unsigned [], int);
7  int cont(unsigned [], int, int, int);
8  void compute_intervals(unsigned [], unsigned, unsigned, int);
9
10
11  //////////////////////////////////////
12  //                               WHERE THE MAGIC HAPPENS
13  //////////////////////////////////////
14
15
16  // Dumb version:
17  // Simply look for the maximum element...
18
19  int dumbMax(unsigned v[], int n) {
20      int max = v[0];
21      for (auto i = 0; i < n; i++)
22          if (max < v[i])
23              max = v[i];
24      return max;
25 }
26
27
28  // Better version:
29  // Use sort() and extract the last element in the sorted array
30
31  int smartMax(unsigned v[], int n) {
32      std::sort(v, v + n);
33      return v[n - 1];
34 }
35
36
37  // This is not very intelligent...
38  // For every single element I will have to scan the entire array.
39  // There are probably better ways to do it.
40
41  int cont(unsigned v[], int m, int M, int n) {
42      int tot = 0;
43      for (auto i = 0; i < n; i++) {
44          if (v[i] >= m && v[i] < M)
45              tot++;
46      }
47      return tot;
48 }
49
50
51  void compute_intervals(unsigned v[], unsigned n, unsigned l, int M) {
52      int k, i;
53      k = ceil((double)M / l);
54      for (i = 0; i < (l - 1); i++)
55          std::cout << i * k << " : " << cont(v, (i * k), ((i + 1) * k), n) << "\n";
56
57      std::cout << i * k << " : " << cont(v, (i * k), (M + 1), n) << "\n";
58 }

```