```cpp
#include <iostream>
#include <vector>
#include <limits>
#include <complex>
#include <sstream>

using namespace std;

bool get_line(const string& prompt, string& userinput){
    cout << prompt;
    getline(cin, userinput);
    return !userinput.empty();
}

bool equal(double x, double y)
{
    return std::fabs(x - y) <= std::numeric_limits<double>
::epsilon() ;
}

double median_of_two_sorted_arrays(const vector<int>& a,
int a_lo, int a_hi,const vector<int>& b, int b_lo, int
b_hi){
    if(a_hi - a_lo <= 1){
        return (max(a[a_lo], b[b_lo]) + min(a[a_hi], b[
b_hi])) / 2.0;
    }

    int a_mid = (a_hi + a_lo) / 2;
    int b_mid = (b_hi + b_lo) / 2;
    bool even_sized = (a_hi - a_lo + 1) % 2 == 0;

    double a_median;
    double b_median;
    if(even_sized){
        a_median = (a[a_mid] + a[a_mid + 1]) / 2.0;
        b_median = (b[b_mid] + b[b_mid + 1]) / 2.0;
    } else {
        a_median = a[a_mid];
        b_median = b[b_mid];
    }

    if(equal(a_median, b_median)){
        return a_median;
    }

    // in the case we have even arrays to ensure valid
splittings
    if(even_sized && a_median < b_median){
        return median_of_two_sorted_arrays(a, a_mid + 1,
```

```
45 a_hi, b, b_lo, b_mid);
46     } else if(even_sized && a_median > b_median) {
47         return median_of_two_sorted_arrays(a, a_lo, a_mid,
    b, b_mid + 1, b_hi);
48     }
49 
50     // in the case of odd arrays to ensure valid
   splittings
51     if(a_median < b_median){
52         return median_of_two_sorted_arrays(a, a_mid, a_hi,
    b, b_lo, b_mid);
53     } else {
54         return median_of_two_sorted_arrays(a, a_lo, a_mid,
    b, b_mid, b_hi);
55     }
56 }
57 
58 double median_of_two_sorted_arrays(const vector<int>& a,
   const vector<int>& b){
59     return median_of_two_sorted_arrays(a, 0, a.size() - 1,
   b, 0, b.size() - 1);
60 }
61 
62 int split_point(const vector<int> &A){
63     int lo = 0, hi = A.size() - 1;
64     int result = -1;
65     while(lo <= hi){
66         int mid = (lo + hi) / 2;
67         if(A[mid] == 0){
68             result = mid;
69             lo = mid + 1;
70         } else{
71             hi = mid - 1;
72         }
73     }
74     return result + 1;
75 }
76 
77 /**
78    example usage for part1:
79    enter a binary where the first k elements are '0' and
   the rest of the n - k elements are '1': 0 0 0 1 1
80    output:  K = 3
81 
82    example usage for part2:
83    press any key then enter to continue: a
84    enter n sorted elements for a1: 0 2 10 26 68
85    enter n sorted elements for a2: 1 11 18 20 41
86    the median of a1 and a2 is : 14.5
87    press any key then enter to continue: a
```

```
 88    enter n sorted elements for a1: 5 6 14 26
 89    enter n sorted elements for a2: 3 41 88 100
 90    the median of a1 and a2 is : 20
 91    press any key then enter to continue: d
 92    enter n sorted elements for a1: 5 10
 93    enter n sorted elements for a2: 2 41
 94    the median of a1 and a2 is : 7.5
 95    press any key then enter to continue:
 96  */
 97 int main(){
 98     string userinput;
 99
100     while(get_line("(part 1) enter a binary where the
    first k elements are '0' and the rest of the n - k
    elements are '1': ", userinput)){
101         vector<int> binary_array;
102         stringstream ss(userinput);
103         int value;
104         while(ss >> value){
105             binary_array.push_back(value);
106         }
107         int k = split_point(binary_array);
108         cout << "K = " << k << endl;
109     }
110
111     while(get_line("(part 2) press any key then enter to
    continue: ", userinput)){
112         cout << "enter n sorted elements for a1: ";
113         getline(cin, userinput);
114         stringstream ss(userinput);
115         vector<int> a1;
116         int value;
117         while(ss >> value){
118             a1.push_back(value);
119         }
120         cout << "enter n sorted elements for a2: ";
121         getline(cin, userinput);
122         ss = stringstream(userinput);
123         vector<int> a2;
124         while(ss >> value){
125             a2.push_back(value);
126         }
127         cout << "the median of a1 and a2 is : " <<
    median_of_two_sorted_arrays(a1, a2)  << endl;
128     }
129 }
130
131
```