```
#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_h
i){
   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, a_hi, b, b_lo, b_mid);
   } else if(even_sized && a_median > b_median) {
      return median_of_two_sorted_arrays(a, a_lo, a_mid, b, b_mid + 1, b_hi);
   }

   // in the case of odd arrays to ensure valid splittings
   if(a_median < b_median){
      return median_of_two_sorted_arrays(a, a_mid, a_hi, b, b_lo, b_mid);
   } else {
```

```
        return median_of_two_sorted_arrays(a, a_lo, a_mid, b, b_mid, b_hi);
    }
}

double median_of_two_sorted_arrays(const vector<int>& a, const vector<int>& b){
    return median_of_two_sorted_arrays(a, 0, a.size() - 1,b, 0, b.size() - 1);
}

int split_point(const vector<int> &A){
    int lo = 0, hi = A.size() - 1;
    int result = -1;
    while(lo <= hi){
        int mid = (lo + hi) / 2;
        if(A[mid] == 0){
            result = mid;
            lo = mid + 1;
        } else{
            hi = mid - 1;
        }
    }
    return result + 1;
}

/**
   example usage for part1:
   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
   output:  K = 3

   example usage for part2:
   press any key then enter to continue: a
   enter n sorted elements for a1: 0 2 10 26 68
   enter n sorted elements for a2: 1 11 18 20 41
   the median of a1 and a2 is : 14.5
   press any key then enter to continue: a
   enter n sorted elements for a1: 5 6 14 26
   enter n sorted elements for a2: 3 41 88 100
   the median of a1 and a2 is : 20
   press any key then enter to continue: d
   enter n sorted elements for a1: 5 10
   enter n sorted elements for a2: 2 41
   the median of a1 and a2 is : 7.5
   press any key then enter to continue:
 */
int main(){
    string userinput;

    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)){
        vector<int> binary_array;
        stringstream ss(userinput);
        int value;
        while(ss >> value){
            binary_array.push_back(value);
        }
```

```
    int k = split_point(binary_array);
    cout << "K = " << k << endl;
  }

  while(get_line("(part 2) press any key then enter to continue: ", userinput)){
    cout << "enter n sorted elements for a1: ";
    getline(cin, userinput);
    stringstream ss(userinput);
    vector<int> a1;
    int value;
    while(ss >> value){
      a1.push_back(value);
    }
    cout << "enter n sorted elements for a2: ";
    getline(cin, userinput);
    ss = stringstream(userinput);
    vector<int> a2;
    while(ss >> value){
      a2.push_back(value);
    }
    cout << "the median of a1 and a2 is : " << median_of_two_sorted_arrays(a1, a2)  << endl;
  }
}
```