```cpp
#include <iostream>
#include <vector>
#include <limits>
#include <random>
#include <chrono>

using namespace std;

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

void display_arr(const vector<int>& arr){
    for(int e : arr) cout << e << " ";
    cout << endl;
}

int find_max_crossing_subarray(const vector<int>& arr, int
 low, int mid, int high){
    int left_sum = std::numeric_limits<int>::min();
    int sum = 0;
    for(int i = mid;i >= low;i--){
        sum += arr[i];
        if(sum > left_sum){
            left_sum = sum;
        }
    }
    int right_sum = std::numeric_limits<int>::min();
    sum = 0;
    for(int j = mid + 1;j <= high;j++){
        sum += arr[j];
        if(sum > right_sum){
            right_sum = sum;
        }
    }
    return left_sum + right_sum;
}

int find_maximum_subarray(const vector<int>& arr, int lo,
int hi){
    if(lo == hi)
        return arr[lo];
    int mid = (lo + hi) / 2;
    int mss_l = find_maximum_subarray(arr, lo, mid);
    int mss_r = find_maximum_subarray(arr, mid + 1, hi);
    int mss_m = find_max_crossing_subarray(arr, lo, mid,
hi);
    return max(mss_l, max(mss_r, mss_m));
}

// O(n * log(n))
int divide_and_conquer_approach(const vector<int> &arr){
    return find_maximum_subarray(arr, 0, arr.size() - 1);
}

// O(n)
int kadane(const vector<int> &arr){
    int max_sum = 0;
    int sum = 0;
    for(int i = 0;i < arr.size(); i++){
        sum += arr[i];
        if(sum > max_sum)
            max_sum = sum;
        else if(sum < 0)
            sum = 0;
    }
    return max_sum;
}

int main() {
    string userinput;
    unsigned seed = chrono::steady_clock::now().time_since_epoch().count();
    mt19937 gen(seed);
    const int LOWER_BOUND = -100;
    const int UPPER_BOUND = 100;
    uniform_int_distribution<int> uniform_int_distribution(LOWER_BOUND, UPPER_BOUND);
    while(get_line("Enter a positive integer: ", userinput)){
        int n = stoi(userinput);
        vector<int> arr;
        for(int i = 0;i < n;i++)
            arr.push_back(uniform_int_distribution(gen));

        display_arr(arr);
        cout << "divide and conquer version: " << divide_and_conquer_approach(arr) << endl;
        cout << "kadane's version: " << kadane(arr) << endl;
    }
}
```