|  |  |
| --- | --- |
| **Max Avg. Subarray in C++** | |
| #include <iostream>  #include <vector>  using namespace std;  double solution(vector<int>& nums, int k) {  int sum = 0;  for (int i = 0; i < k; i++) {  sum += nums[i];  }  int max\_sum = sum;  for (int i = k; i < nums.size(); i++) {  sum += nums[i];  sum -= nums[i - k];  max\_sum = max(max\_sum, sum);  }  return static\_cast<double>(max\_sum) / k;  }  int main() {  vector<int> nums = {-10, 5, -6, 8, -7, 2, -4, 8, -6, 7};  int k = 3;  cout << solution(nums, k) << endl;  return 0;  } | Input: nums = {-10, 5, -6, 8, -7, 2, -4, 8, -6, 7}  k = 3 🔍 Dry Run Table: We'll track the sum of every window of size 3:   | **Window (Indexes)** | **Elements** | **Window Sum** | **max\_sum** | | --- | --- | --- | --- | | 0–2 | -10, 5, -6 | -11 | -11 | | 1–3 | 5, -6, 8 | 7 | 7 | | 2–4 | -6, 8, -7 | -5 | 7 | | 3–5 | 8, -7, 2 | 3 | 7 | | 4–6 | -7, 2, -4 | -9 | 7 | | 5–7 | 2, -4, 8 | 6 | 7 | | 6–8 | -4, 8, -6 | -2 | 7 | | 7–9 | 8, -6, 7 | 9 | **9** |  ✅ Final Output: 9 / 3 = 3.0  ✔️ Output: 3 |
| 3 | |