Find K closest elements in C++

<table>
<tr><td>

```cpp
#include <iostream>
#include <vector>
#include <cstdlib> // for abs function
#include <algorithm> // for sort function
using namespace std;

class FindKClosestElements {
public:
    static vector<int>
findClosest(vector<int>& arr, int k, int x)
{
    int lo = 0;
    int hi = arr.size() - 1;

    // Using binary search to find the
range of k closest elements
    while (hi - lo >= k) {
       if (abs(arr[lo] - x) > abs(arr[hi] -
x)) {
          lo++;
       } else {
          hi--;
       }
    }

    // Extract the k closest elements into
a vector
    vector<int> result(arr.begin() + lo,
arr.begin() + lo + k);

    return result;
    }
};

int main() {
   // Hardcoded input
   vector<int> arr = {10, 20, 30, 40, 50,
60};
   int k = 3;
   int x = 45;

   // Call the findClosest function to find k
closest elements to x
   vector<int> ans =
FindKClosestElements::findClosest(arr,
k, x);

   // Print the closest elements
   cout << "Closest elements to " << x <<
": ";
   for (int val : ans) {
      cout << val << " ";
   }
   cout << endl;

   return 0;
}
```

</td><td>

Here's a **detailed tabular dry run** of your code using the input:

```
arr = {10, 20, 30, 40, 50, 60}
k = 3
x = 45
```

🔴 **Goal:**

Find the **k = 3** elements in arr that are **closest to x = 45** using the two-pointer approach.

Initial Setup:

- lo = 0, hi = 5 (last index)
- Keep shrinking the window from either end until hi - lo + 1 == k

🔍 **Step-by-Step Table:**

Step	lo	hi	hi - lo	abs(arr[lo] - x)	abs(arr[hi] - x)	Decision	New lo	New hi
1	0	5	5	abs(10 - 45) = 35	abs(60 - 45) = 15	35 > 15 → shrink left	1	5
2	1	5	4	abs(20 - 45) = 25	abs(60 - 45) = 15	25 > 15 → shrink left	2	5
3	2	5	3	abs(30 - 45) = 15	abs(60 - 45) = 15	Equal → shrink right	2	4

Now, hi - lo + 1 = 3, so stop.

✔️ **Final Window:**

arr[2] to arr[4] → {30, 40, 50}

Closest elements to 45 are:

30 40 50

🖥️ **Final Output:**

Closest elements to 45: 30 40 50

</td></tr>
</table>

Closest elements to 45: 30 40 50