K closest elements

> 给定数组arr与整数X，找到arr中离整数X位置上最近的K个数

采用二分首先找到X的位置，然后从X开始，定义xleft，xright指针，不断向两边移动，直到满足K个数

```
#include <bits/stdc++.h>

using namespace std;

int binarySearch(vector<int> &arr, int left, int right, int target) {
    if (arr[right] <= target) return right;
    if (arr[left] > target) return left;
    int mid = left + (right - left) / 2;
    if (arr[mid] <= target && arr[mid + 1] > target) return mid;
    if (arr[mid] < target) return binarySearch(arr, mid + 1, right, target);
    return binarySearch(arr, left, mid - 1, target);
}

void kClosestElements(vector<int> &arr, int &N, int &K, int &X) {
    int xleft = binarySearch(arr, 0, N - 1, X);
    int xright = xleft + 1;
    int count = 0;

    // 如果X本身就在数组当中，则Arr[xleft]实际上就是X，所以需要左移 xleft
    if (arr[xleft] == X) xleft--;

    // xlefts存储X左边的,xrights存储X右边的
    vector<int> xlefts, xrights;
    while (xleft >= 0 && xright < N && count < K) {
        xlefts.push_back(arr[xleft--]);
        xrights.push_back(arr[xright++]);
        count += 2;
    }

    while (count < K && xleft >= 0) {
        xlefts.push_back(arr[xleft--]);
        count++;
    }
    while (count < K && xright < N) {
        xrights.push_back(arr[xright++]);
        count++;
    }
    // 注意倒置xlefts
    reverse(xlefts.begin(), xlefts.end());
    for (int l: xlefts)
        printf("%d ", l);
    for (int r: xrights) {
        printf("%d ", r);
    }
}
```

```
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N, num, K, X;
        scanf("%d", &N);
        vector<int> arr;
        for (int i = 0; i < N; ++i) {
            scanf("%d", &num);
            arr.push_back(num);
        }
        scanf("%d %d", &K, &X);
        kClosestElements(arr, N, K, X);
        printf("\n");
    }
    return 0;
}
```