```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;


int QuickSelect(int[], int, int, int);
int Partition(int[], int,int);
int QuickSelectMax(int[],int,int,int);
int main()
{       int n,k,j=999;
        cout<<"Enter positive integer n to generate n random integers"<<endl;
        cin >> n;

        int *arr= new int[n];
        for (int i=0; i<n; i++)
        {
                int num= rand() %200 +(-100);
                arr[i]=num;
        }
        for (int j=0; j<n; j++)
                cout<<arr[j]<<", ";
        cout<<endl<<"Enter number k to find k least element"<<endl;
        cin >>k;

        cout<<QuickSelect(arr,k,0,n)<<endl;

        int a[]={4,2,0,10,16,3,7,8,38,86};


        cout<<"Enter k to find max k numbers"<<endl;
        cin>>j;
        cout<<QuickSelectMax(arr,j,0,n-1)<<endl;


}
int QuickSelectMax(int arr[], int k,int begin,int end) //Find Max K elements
{
        int part= Partition(arr,begin,end); //partition
        int diff = end-part; //difference between partition pointer and end of
       //       array

        if(k==end+1)
        {
                for(int i=0; i<end;i++)
                        cout<<arr[i]<<" ";
                cout<<endl<<"Displaying Max: ";
                return k;
        }
        if(diff==k) //if the numbers to the right of the partition equals k
                   // we know they will be bigger and be value looking for
        {       for(int i =part+1; i<= end; i++)
                {       cout<<arr[i]<<" ";

                }
                cout<<endl<<"Displaying Max: ";
                return k;
        }
        if(diff<k) // if difference is less than k then we have to few values
                  // repartition again to get correct range and values
                return QuickSelectMax(arr,k,part-1,end);
        return QuickSelectMax(arr,k,part+1,end); // if differcene is greater
                                                // then pointer is too low,
                                                // move partition pointer up
                                                // one and repartition

}



int QuickSelect(int arr[], int k,int begin,int end)
{
        int part= Partition(arr,begin,end);
        if(part==k-1)                           //if only one
                return (arr[part]);
        if(part>k)                              //if pivot index > k search left array
                return QuickSelect(arr,k,0,part);
        return QuickSelect(arr,k, part+1,end); //if pivot< index search right array
adjust k value K-partition

}


int Partition(int arr[], int start, int end)
{
        int pivot=arr[start];
        int finish=end;
        int begin=start;
        int findPart=0;
        while(finish>begin)
        {
        while(arr[begin]<pivot)
                begin+=1;
        while(arr[finish]>pivot)
                finish-=1;
        if(finish==begin)
                break;

        swap(arr[begin],arr[finish]);
        }

        while(arr[findPart]!=pivot)
                findPart+=1;            //return the pivot index
        return findPart;

}
```