单调队列

给定一个大小为$n \leq 10^6$的数组。

有一个大小为k的滑动窗口，它从数组的最左边移动到最右边。

您只能在窗口中看到k个数字。

每次滑动窗口向右移动一个位置。

以下是一个例子：

该数组为[1 3 -1 -3 5 3 6 7]，k为3。

窗口位置	最小值	最大值
[1 3 -1] -3 5 3 6 7	-1	3
1 [3 -1 -3] 5 3 6 7	-3	3
1 3 [-1 -3 5] 3 6 7	-3	5
1 3 -1 [-3 5 3] 6 7	-3	5
1 3 -1 -3 [5 3 6] 7	3	6
1 3 -1 -3 5 [3 6 7]	3	7

您的任务是确定滑动窗口位于每个位置时，窗口中的最大值和最小值。

输入格式

输入包含两行。

输入格式

输入包含两行。

第一行包含两个整数n和k，分别代表数组长度和滑动窗口的长度。

第二行有n个整数，代表数组的具体数值。

同行数据之间用空格隔开。

输出格式

输出包含两个。

第一行输出，从左至右，每个位置滑动窗口中的最小值。

第二行输出，从左至右，每个位置滑动窗口中的最大值。

输入样例：

```
8 3
1 3 -1 -3 5 3 6 7
```

输出样例：

```
-1 -3 -3 -3 3 3
3 3 5 5 6 7
```

题解思路

我们用q来表示单调队列，p来表示其所对应的在原列表里的序号。

由于此时队中没有一个元素，我们直接令1进队。此时，q={1},p={1}。

现在3面临着抉择。下面基于这样一个思想:假如把3放进去，如果后面2个数都比它大，那么3在其有生之年就有可能成为最小的。此时，q={1,3},p={1,2}

下面出现了-1。队尾元素3比-1大，那么意味着只要-1进队，那么3在其有生之年必定成为不了最小值，原因很明显:因为当下面3被框起来，那么-1也一定被框起来，所以3永远不能当最小值。所以，3从队尾出队。同理，1从队尾出队。最后-1进队，此时q={-1},p={3}

出现-3，同上面分析，-1>-3，-1从队尾出队,-3从队尾进队。q={-3}，p={4}。

出现5，因为5>-3，同第二条分析，5在有生之年还是有希望的，所以5进队。此时，q={-3,5},p={4,5}

出现3。3先与队尾的5比较，3<5，按照第3条的分析，5从队尾出队。3再与-3比较，同第二条分析，3进队。此时，q={-3,3},p={4,6}

出现6。6与3比较，因为3<6，所以3不必出队。由于3以前元素都＜3，所以不必再比较，6进队。因为-3此时已经在滑动窗口之外，所以-3从队首出队。此时，q={3,6},p={6,7}

出现7。队尾元素6小于7，7进队。此时，q={3,6,7},p={6,7,8}。

那么，我们对单调队列的基本操作已经分析完毕。因为单调队列中元素大小单调递*(增/减/自定义比较),
因此，队首元素必定是最值。按题意输出即可。

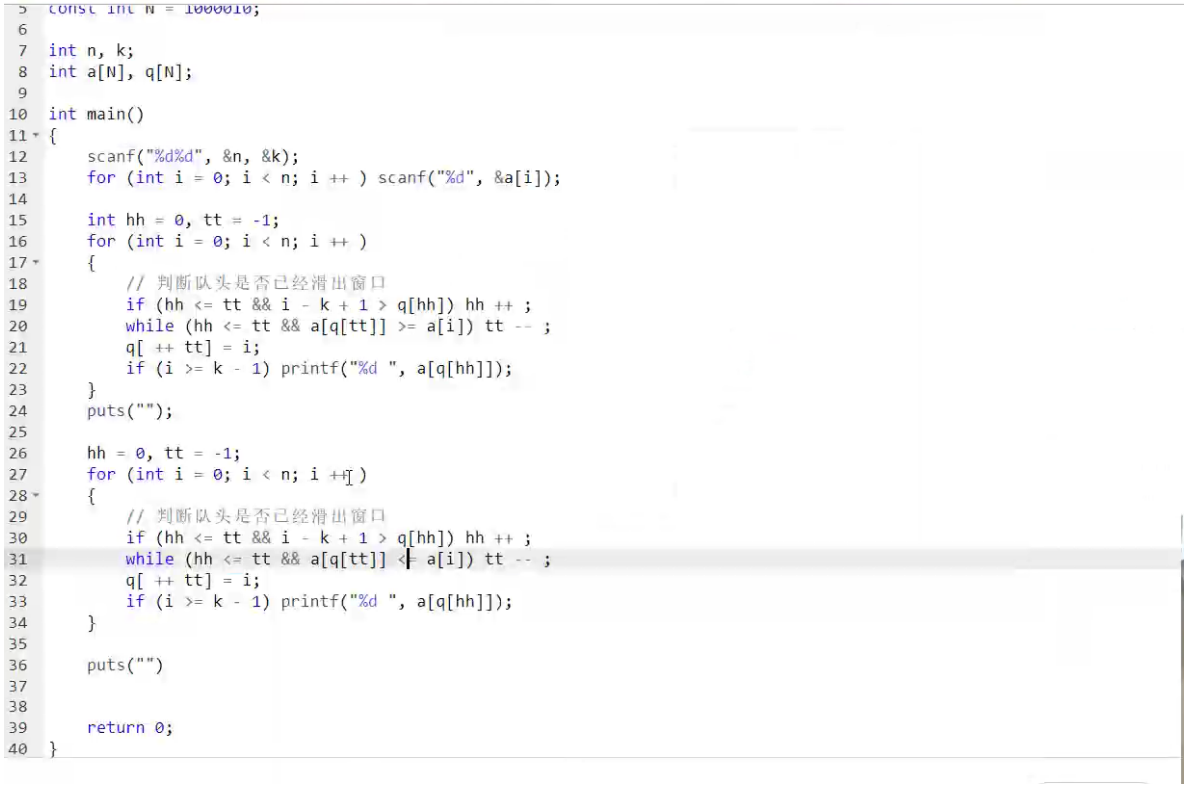
```
#include<bits/stdc++.h>
using namespace std;
template<typename T>inline void read(T &x)
{
    x=0;T f=1,ch=getchar();
    while(!isdigit(ch))  {if(ch=='-')  f=-1;  ch=getchar();}
    while(isdigit(ch))  {x=x*10+ch-'0';  ch=getchar();}
    x*=f;
}
int head,tail,q[1000001],p[1000001],k,n,a[1000001];
inline void maxn()
{
    head=1;tail=0;
    for(int i=1;i<=n;i++)
    {
        while(head<=tail&&q[tail]<=a[i])
            --tail;//从队尾出队；
        q[++tail]=a[i];//入队；
        p[tail]=i;//记录在原序列位置；
        while(p[head]<=i-k)//长度不超过k；
            head++;
        if(i>=k) printf("%d ",q[head]);
    }
    putchar('\n');
}
inline void minn()
{
    head=1;tail=0;
    for(int i=1;i<=n;i++)
    {
        while(head<=tail&&q[tail]>=a[i])
```

```
            --tail;//只要队列里有元素，并且尾元素比待处理值大，即表示尾元素已经不可能成为最小值，所以出队。直到尾元素小于待处理值，满足"单调"。
        q[++tail]=a[i];
        p[tail]=i;
        while(p[head]<=i-k)
            head++;
        if(i>=k) printf("%d ",q[head]);//满足题意输出
    }
    putchar('\n');
}
int main()
{
    read(n);read(k);
    for(int i=1;i<=n;i++)
        read(a[i]);
    minn();
    maxn();
    return 0;
}
```

作者：Tyouchie
链接：https://www.acwing.com/solution/content/2499/
来源：AcWing
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

作者：Tyouchie
链接：https://www.acwing.com/solution/content/2499/
来源：AcWing
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

y总代码：



```
const int N = 1000010;

int n, k;
int a[N], q[N];

int main()
{
    scanf("%d%d", &n, &k);
    for (int i = 0; i < n; i ++ ) scanf("%d", &a[i]);

    int hh = 0, tt = -1;
    for (int i = 0; i < n; i ++ )
    {
        // 判断队头是否已经滑出窗口
        if (hh <= tt && i - k + 1 > q[hh]) hh ++ ;
        while (hh <= tt && a[q[tt]] >= a[i]) tt -- ;
        q[ ++ tt] = i;
        if (i >= k - 1) printf("%d ", a[q[hh]]);
    }
    puts("");

    hh = 0, tt = -1;
    for (int i = 0; i < n; i ++ )
    {
        // 判断队头是否已经滑出窗口
        if (hh <= tt && i - k + 1 > q[hh]) hh ++ ;
        while (hh <= tt && a[q[tt]] <= a[i]) tt -- ;
        q[ ++ tt] = i;
        if (i >= k - 1) printf("%d ", a[q[hh]]);
    }

    puts("")


    return 0;
}
```

```
#include <iostream>

using namespace std;
```

```cpp
const int N = 1000010;

int a[N], q[N];

int main()
{
    int n, k;
    scanf("%d%d", &n, &k);
    for (int i = 0; i < n; i ++ ) scanf("%d", &a[i]);

    int hh = 0, tt = -1;
    for (int i = 0; i < n; i ++ )
    {
        if (hh <= tt && i - k + 1 > q[hh]) hh ++ ;

        while (hh <= tt && a[q[tt]] >= a[i]) tt -- ;
        q[ ++ tt] = i;

        if (i >= k - 1) printf("%d ", a[q[hh]]);
    }

    puts("");

    hh = 0, tt = -1;
    for (int i = 0; i < n; i ++ )
    {
        if (hh <= tt && i - k + 1 > q[hh]) hh ++ ;

        while (hh <= tt && a[q[tt]] <= a[i]) tt -- ;
        q[ ++ tt] = i;

        if (i >= k - 1) printf("%d ", a[q[hh]]);
    }

    puts("");

    return 0;
}
```