分析：很明显的一道单调队列，为什么呢？后来加入队伍的人（注意是题目的队伍而不是代码中的队列，后文也要注意区别队伍和队列），如果通过查找把它插到递减队列中对应的位置，而不用维护其后面的值。因为在这个新人出队之前，在它前面来的人（准确的说是RP更低的被抛弃的那些人）是不可能成为Q的结果的，也就不用再维护了。

同时我们记录一个总出现人数n和已经面试完的人数out，每次Q如果out==n，说明没有人排队，输出-1，否则输出队首元素，因为此时是RP最高的人。对于G，如果out的人已经波及到队首元素的pos，即out>pos[queue[l]]，那么说明RP最高者已经不是队首了，因为他已经out，则队首向前进发即l++ 。

我们单调队列不是直接在你读入的存值的数组上进行操作的，而是要另开一个数组，并且维护的是下标

本题出队顺序用一个变量out来记录就可以，因为顺序出栈的，每出一个out++

#include<cstdio>

#include<iostream>

using namespace std;

typedef long long ll;

const int INF=1000000007;

ll a[1000010];

char s[10];

int q[1000010];

int main()

{

// freopen("input.txt","r",stdin);

int T;

scanf("%d",&T);

while(T--)

{

scanf("%s",s);

int n(0);

int l(1),r(0),out(0);

while(~scanf("%s",s))

{

if(s[0]=='E')

break;

if(s[0]=='C')

{

scanf("%s",s);

scanf("%lld",&a[++n]);

while(l<=r && a[q[r]]<a[n])

r--;

q[++r]=n;

}

else if(s[0]=='G')

{

out++;

while(q[l]<=out)l++;

}

else if(s[0]=='Q')

{

if(out>=n)

printf("-1\n");

else

{

printf("%lld\n",a[q[l]]);

}

}

}

}

return 0;

}