#include<iostream>

#include"max.h"

#include"string"

#include"math.h"

#include"vector"

#include"map"

#include <unordered\_set>

#include <algorithm>

#include <cstdlib>

#include <sstream>

using namespace std;

//下面的main函数请勿修改

#include <stdio.h>

#include <stdlib.h>

struct node

{//链表结点类型，包含一个存放整型数据的 data 成员，和一个指向下一个结点的next成员

    int data ;

    struct node \*next ;

};

//第一关代码

struct node \*createRlist()

{//函数功能：创建一个有一个空循环链表，返回值为头指针

    struct node \* list=(struct node \*)malloc(sizeof(struct node));

    list->data=0;

    list->next=list;

    return list;

}

struct node \* insertOrder(struct node \*list, int insData)

{

 //在单向递增有序的循环链表（表头指针list）中插入数据元素insData，使之依然有序 。返回值是头指针

     struct node \* p=(struct node \*)malloc(sizeof(struct node));

     p->data=insData;

     p->next=NULL;

     if(list->next==list){//保证list 指针用于指向最大的节点

         list->next=p;

        p->next=list;

       list = p;

        return list;

     }

     struct node \* head=list->next;

     struct node \*pre=head;

     struct node \* cur=pre->next;

        while(cur!=head&&cur->data<insData){

              pre=cur;

              cur=cur->next;

         }

         p->next=cur;

         pre->next=p;

         if(head==cur) {//保证list 指针用于指向最大的节点 更改list 指针

            list=p;

         }

 return list;

}

int deleteData(struct node  \*list, int delData)

{

    //在单向递增有序循环链表（表头指针list）中删除所有值为delData的结点,返回值为删除结点的个数

    int sum=0;

    struct node \*head=list->next;

    struct node \* pre=head;

    struct node \*cur=head->next;

    while(cur!=head){//删除操作

      if(cur->data==delData){

            pre->next=cur->next;

            struct node \* op=cur;

            cur=cur->next;

            free(op);

            sum++;

      }

      else{

        pre=cur;

        cur=cur->next;

      }

    }

     return sum;

}

void printRlist(struct node \*list)

{

 //从链表第一个结点开始输出单向循环链表中各数据元素的值，每输出一个数据元素空一格

    struct node \* head=list->next;

    struct node \* cur=head->next;

    while(cur!=head){

        printf("%d",cur->data);

        cur=cur->next;

    }

}

int destroyRlist(struct node \*list)

{

    //从第一个结点开始释放循环链表各结点占用的空间,返回值为最后一个结点的值

    int n=list->data;

    struct node \* head=list->next;

    struct node \* pre=head->next;

    list->next=NULL;

    list=head;

    head->next=NULL;

    struct node \*cur=pre->next;

    head->next=NULL;

    while(cur!=NULL){

      struct node \* tem=pre;

      pre=cur;

      cur=cur->next;

      free(tem);

    }

    return n;

}

int main(void)

{

  struct node \*list=createRlist();

  for(int i=0;i<6;i++){

    int n=0;

    scanf("%d",&n);

    list=insertOrder(list, n);

  }

printRlist(list);

int p=0;

scanf("%d",&p);

 int sum=deleteData(list,p);

 printRlist(list);

 printf("   ans:%d ",sum);

 int ans=destroyRlist(list);

 printf("%d",ans);

printRlist(list);

}