**盾神与条状项链**

问题描述

　　有一天，盾神捡到了好多好多五颜六色的珠子！他心想这些珠子这么漂亮，可以做成一条项链然后送给他心仪的女生~于是他用其中一些珠子做成了长度为n的项链。当他准备把项链首尾相接的时候，土方进来了。  
　　“哇这么恶心的项链你也做得出来！！！”  
　　盾神自知审美不是他的长项，于是他很谦虚地请教土方，怎么才能把项链做得漂亮。  
　　“这个嘛~首先你要在这里加上一个这种颜色的珠子，然后在这里去掉这个珠子，然后……，最后你看看是不是漂亮很多咧~”土方一下子说出了m个修改步骤。  
　　盾神觉得这个用人工做太麻烦了，于是交给了你。

输入格式

　　第一行两个数，分别为n，m。  
　　第二行n个数，表示盾神一开始的项链。第i个数表示第i颗珠子的颜色。  
　　接下来m行，为以下形式之一：  
　　ADD P Q：表示在颜色为P的珠子前面加上一个颜色为Q的珠子。  
　　DEL P：表示把颜色为P的珠子去掉，如果它不在端点处，则需要把它旁边的两颗珠子连起来。例如某时刻项链状态为1 4 5 8，则执行DEL 4会变成1 5 8，执行DEL 1会变成4 5 8。  
　　输入保证在每次操作之前，项链有颜色为P的珠子，且任意时刻珠子颜色互不相同。

输出格式

　　第一行为一个数len，为做完所有操作后，项链的长度。  
　　第二行len个数，表示此时项链的状态。第i个数表示第i颗珠子的颜色。

样例输入

10 5  
1 2 3 4 5 6 7 8 9 10  
DEL 5  
ADD 7 5  
DEL 10  
ADD 4 20  
ADD 20 12

样例输出

11  
1 2 3 12 20 4 6 5 7 8 9

数据规模和约定

　　表示颜色的数字不超过10^5的正数，1<=n<=10^4，1<=m<=10^4。

C++代码：

#include <stdio.h>

#include <stdlib.h>

int n,m;

struct u

{

int color;

struct u \* next;

struct u \* front;

};

struct u \* p[100001];

struct u \*head;

int main()

{

int i;

scanf("%d%d",&n,&m);

for(i=1;i<=n;i++)

{

int a;

scanf("%d",&a);

struct u \*q=(struct u \*)malloc(sizeof(struct u));

q->color=a;

q->next=head;

p[a]=q;

head=q;

if(head->next!=NULL)

{

head->next->front=head;

}

}

head->front=NULL;

for(i=1;i<=m;i++)

{

char s[4];

scanf("%s",s);

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

{

n--;

int x;

scanf("%d",&x);

struct u \*t1=p[x]->next;

struct u \*t2=p[x]->front;

if(t2==NULL)

{

t1->front=NULL;

head=t1;

}

else if(t1==NULL)

{

t2->next=NULL;

p[x]=NULL;

}

else

{

t2->next=t1;

t1->front=t2;

}

p[x]=NULL;

}

else

{

n++;

int x,y;

scanf("%d%d",&x,&y);

struct u \*t1=p[x]->next;

p[y]=(struct u \*)malloc(sizeof(struct u));

p[x]->next=p[y];

p[y]->front=p[x];

p[y]->next=t1;

if(t1!=NULL) t1->front=p[y];

p[y]->color=y;

}

}

struct u \*t=head;

while(t->next!=NULL)

{

t=t->next;

}

printf("%d\n",n);

while(t!=NULL)

{

printf("%d ",t->color);

t=t->front;

}

return 0;

}//walk slowly but never stop

C代码：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAX 30000000

typedef struct ball

{

long left, right;

}Ball;

Ball \*balls;

void add(int previous, int curr)

{

balls[curr].right = balls[previous].right;

balls[curr].left = previous;

balls[previous].right = curr;

}

void del(int index)

{

balls[balls[index].left].right = balls[index].right;

if (balls[index].right != -1)

balls[balls[index].right].left = balls[index].left;

balls[index].left = -1;

balls[index].right = -1;

}

void add\_one(int index, int new\_one)

{

balls[new\_one].right = index;

balls[new\_one].left = balls[index].left;

balls[balls[index].left].right = new\_one;

balls[index].left = new\_one;

}

int main(void)

{

balls = (Ball \*)malloc(sizeof(Ball) \* MAX);

memset(balls, -1, sizeof(Ball)\*MAX);

int n, m, len, input,op1,op2;

char op[10];

scanf("%d %d", &n, &m);

int previous = 0;

int count = n;

while(n--)

{

scanf("%d", &input);

add(previous, input);

previous = input;

}

while(m --)

{

scanf("%s",op);

if (op[0] == 'D')

{

scanf("%d", &op1);

del(op1);

count --;

}

else

{

scanf("%d%d",&op1,&op2);

add\_one(op1,op2);

count ++;

}

}

printf("%d\n",count);

int i = balls[0].right;

while(balls[i].right != -1)

{

printf("%d ", i);

i = balls[i].right;

}

printf("%d",i);

return 0;

}

JAVA代码：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.io.StreamTokenizer;

import java.util.ArrayList;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws IOException {

StreamTokenizer in = new StreamTokenizer(new BufferedReader(new InputStreamReader(System.in)));

PrintWriter out = new PrintWriter(new OutputStreamWriter(System.out));

ArrayList<Integer> list = new ArrayList<Integer>();

in.nextToken();

int n = (int) in.nval;

in.nextToken();

int m = (int) in.nval;

for(int i = 0;i<n;i++)

{

in.nextToken();

list.add((int)in.nval);

}

for(int i = 0;i<m;i++)

{

in.nextToken();

String st = in.sval;

in.nextToken();

int num =(int) in.nval;

if(st.equals("DEL"))

{

int p = list.indexOf(num);

list.remove(p);

}

else if(st.equals("ADD"))

{

in.nextToken();

int num2 = (int) in.nval;

int w = list.indexOf(num);

list.add(w, num2);

}

}

out.println(list.size());

for(int i = 0;i<list.size()-1;i++)

out.print(list.get(i)+" ");

out.println(list.get(list.size()-1));

out.flush();

}

}