汉诺塔

方法一：类

```
#include<iostream>
#include<string>
#include<cstring>
using namespace std;
#define TRUE    1
#define FALSE   0
#define OK      1
#define ERROR   0
#define LOVERFLOW   -1
typedef int SElemType;
typedef  int Status;
#define  STACK_INIT_SIZE  1000   //初始大小为1000
#define  STACKINCREMENT  10   //若空间不够，每次增长10
class  SqStack
{
protected:
    SElemType *base;
    SElemType *top;
public:
    int   stacksize;
    int   count;
    SqStack();
    ~SqStack();
    Status StackEmpty();
    Status GetTop();
    Status Push(SElemType);
    Status Pop(SElemType&);
    void friend MOVE_Disk(SqStack CS[], int, int);
    void friend Display(SqStack CS[], int);
};
SqStack::SqStack()
{
    base = new SElemType[STACK_INIT_SIZE];
    if (base == NULL)
        exit(LOVERFLOW);
    top = base;
    stacksize = 0;
    count = 0;
}
SqStack::~SqStack()
{
    if (base)
        delete base;
    top = NULL;
    stacksize = 0;
}
Status SqStack::StackEmpty()
{
    if (top == base)
        return TRUE;
    else
        return FALSE;
}
Status SqStack::GetTop()
{
    if (count == 0)
        return 0;
    else
        return  *(top - 1);
}
Status SqStack::Push(SElemType e)
{
    /*如果栈已满则扩充空间*/
    if (top - base >= stacksize)
    {
        SElemType *newbase;
        newbase = new SElemType[stacksize + STACK_INIT_SIZE];
        if (!newbase)
            return LOVERFLOW;
        memcpy(newbase, base, stacksize * sizeof(SElemType));
        delete base;
        base = newbase;
        top = base + stacksize;
        stacksize += STACKINCREMENT;
    }
    *top++ = e;
    count++;
    return OK;
}
Status SqStack::Pop(SElemType &e)
{
    if (top == base)
        return ERROR;
    e = *--top;
    count--;
    return OK;
}
void MOVE_Disk(SqStack CS[], int a, int b)
{
    int c;
    int flag = 1;
    if (CS[a - 1].StackEmpty())
    {
        cout << a << " " << "IS EMPTY" << endl;
        flag = 0;
    }
    if (CS[b - 1].count == CS[b - 1].stacksize)
    {
        cout << b << " " << "IS FULL" << endl;
        flag = 0;
    }
    if (CS[a - 1].GetTop() > CS[b - 1].GetTop() && !CS[b -
1].StackEmpty())
    {
        cout << "ILLEGAL" << endl;
        flag = 0;
    }
    if (flag == 1)
    {
        cout << CS[a - 1].GetTop() << endl;
        CS[a - 1].Pop(c);
        CS[b - 1].Push(c);
    }
}
void Display(SqStack CS[], int a)
{
    if (!CS[a - 1].StackEmpty())
    {
        int *p = CS[a - 1].top - 1;
        while (p >= CS[a - 1].base)
        {
            cout << *p << " ";
            p--;
        }
        cout << endl;
    }
    else
        cout << 0 << endl;
}
```

```cpp
void PrintCS(SqStack CS[], int n)
{
    int i = 0, c;
    for (i = 0; i < n; i++)
    {
        if (!CS[i].StackEmpty())
            while (!CS[i].StackEmpty())
            {
                CS[i].Pop(c);
                cout << c << " ";
            }
        else
            cout << 0;
        cout << endl;
    }
}
int main()
{
    SqStack CS[STACK_INIT_SIZE];
    int n, k;
    cin >> n >> k;
    int i;
    for (i = 0; i < n; i++)
        cin >> CS[i].stacksize;
    int disk[STACK_INIT_SIZE];
    for (int j = 0; j < k; j++)
    {
        cin >> disk[j];
        CS[0].Push(disk[j]);
    }
    string vis[STACK_INIT_SIZE];
    string s;
    int dis[STACK_INIT_SIZE];
    int a, b, c;
    int x[STACK_INIT_SIZE], y[STACK_INIT_SIZE];
    i = 0;
    int j = 0;
    while (1)
    {
        cin >> s;
        if (s == "MOVE")
        {
            cin >> a >> b;
            vis[i] = "MOVE";
            x[i] = a;
            y[i] = b;
        }
        else if (s == "DISPLAY")
        {
            cin >> c;
            vis[i] = "DISPLAY";
            dis[j++] = c;
        }
        else
        {
            vis[i] = "QUIT";
            break;
        }
        i++;
    }
    i = 0;
    j = 0;
    while (vis[i] != "QUIT")
    {
        if (vis[i] == "MOVE")
            MOVE_Disk(CS, x[i], y[i]);
        else if (vis[i] == "DISPLAY")
        {
            Display(CS, dis[j]);
            j++;
        }
        i++;
    }
    PrintCS(CS, n);
    system("pause");
    return 0;
}
```

方法二：结构体

```
#include<iostream>
#include<string>
#define MAX_SIZE 1000
using namespace std;
struct ColStack
{
    int *top;
    int *base;
    int size;
    int count;
};
ColStack CS[MAX_SIZE];
void IniCS(ColStack cs[], int n)
{
    for (int i = 0; i < n; i++)
    {
        cs[i].base = new int[MAX_SIZE];
        cs[i].top = cs[i].base;
        cs[i].count = 0;
    }
}
void Destroy(ColStack CS[], int n)
{
    for (int i = 0; i < n; i++)
    {
        if (CS[i].base)
            delete CS[i].base;
        CS[i].top = NULL;
    }
}
void Push(ColStack &cs, int x)
{
    *cs.top++ = x;
    cs.count++;
}
void Pop(ColStack &cs, int&x)
{
    x = *--cs.top;
    cs.count--;
}
bool Full(ColStack &cs)
{
    if (cs.count >= cs.size)
        return true;
    return false;
}
bool Empty(ColStack &cs)
{
    if (cs.top == cs.base)
        return true;
    return false;
}
void MOVE_Disk(ColStack CS[], int a, int b)
```

```
{
    int c;
    int flag = 1;
    if (Empty(CS[a - 1]))
    {
        cout << a << " " << "IS EMPTY" << endl;
        flag = 0;
    }
    if (Full(CS[b - 1]))
    {
        cout << b << " " << "IS FULL" << endl;
        flag = 0;
    }
    if (!Empty(CS[b - 1]))
        if (*(CS[a - 1].top - 1) > *(CS[b - 1].top - 1))
        {
            cout << "ILLEGAL" << endl;
            flag = 0;
        }
    if (flag == 1)
    {
        Pop(CS[a - 1], c);
        cout << c << endl;
        Push(CS[b - 1], c);
    }
}
void Display(ColStack CS[], int a)
{
    int x = 0;

    if (!Empty(CS[a - 1]))
    {
        int *p = CS[a - 1].top - 1;
        while (p >= CS[a - 1].base)
        {
            cout << *p << " ";
            p--;
        }
        cout << endl;
    }
    else
    {
        cout << 0 << endl;
    }
```

```cpp
}
void PrintCS(ColStack CS[], int n)
{
    int i = 0, c;
    for (i = 0; i < n; i++)
    {
        if (!Empty(CS[i]))
            while (!Empty(CS[i]))
            {
                Pop(CS[i], c);
                cout << c << " ";
            }
        else
            cout << 0;
        cout << endl;
    }
}
int main()
{
    int n, k;
    cin >> n >> k;
    int i;
    IniCS(CS, n);
    for (i = 0; i < n; i++)
        cin >> CS[i].size;
    int disk[MAX_SIZE];
    for (int j = 0; j < k; j++)
    {
        cin >> disk[j];
        Push(CS[0], disk[j]);
    }
    string vis[MAX_SIZE];
    string s;
    int dis[MAX_SIZE];
    int a, b, c;
    int x[MAX_SIZE], y[MAX_SIZE];
    i = 0;
    int j = 0;
    while (1)
    {
        cin >> s;
        if (s == "MOVE")
        {
            cin >> a >> b;
            vis[i] = "MOVE";
            x[i] = a;
            y[i] = b;
        }
        else if (s == "DISPLAY")
        {
            cin >> c;
            vis[i] = "DISPLAY";
            dis[j++] = c;
        }
        else
        {
            vis[i] = "QUIT";
            break;
        }
        i++;
    }
    i = 0;
    j = 0;
    while (vis[i] != "QUIT")
    {
        if (vis[i] == "MOVE")
            MOVE_Disk(CS, x[i], y[i]);
        else if (vis[i] == "DISPLAY")
        {
            Display(CS, dis[j]);
            j++;
        }
        i++;
    }
    PrintCS(CS, n);
    return 0;
}
```