定义在函数main中的变量需要自己进行初始化

#include<memory.h>

memset(visit, 0, sizeof(visit));

Node\* r = (node\*)mallco(sizeof(node));

scanf用于格式化输入非常好用

scanf("%lld/%lld %lld/%lld", &a, &b, &c, &d);

#include<map>

map<double, char>::reverse\_iterator re\_it = game[i].rbegin();

(\*it).first; it->first;

vector部分排序使用两个迭代器确定具体范围

字符变数字，进制中进行

string n;long long num;

num = n[i] - 'a' ;

数字要转换成字符则需要使用sstream中的stringstream

字符串的大小写转换

#include <string>

#include <algorithm>

#include <iterator>

#include <cctype>

transform(src.begin(), src.end(), back\_inserter(dst), ::toupper);

transform(src.begin(), src.end(), dst.begin(), ::tolower);

printf("%.2lf", it->second);

printf("%.1d", it->second);

switch (i)

{

case 0:

cout << "zero";

break;

default:

break;

}

int INF = 1 << 30;

判断浮点数不为零

if (it->second-0>0.00001||it->second-0<-0.00001)

Prime [数] 素数

与时间相关的，算时间差，时间先后可以通过从一个时间点进行截断，从标志点到这里有多远。

vector<int> lev(100000, -1); 向量开100000个空间全放-1。

%\*% 测试用例可能会输入很大的整型，这时候，用string处理才可以通过。这种题目特征比较明显，题目容易，看似简单，其陷阱在于大整数。

long long是六十四位，因此一部分大数题目可以用long long

还有就是两个大数相加，结果可能越界，如果越界结果会符号变反。可以利用这一性质来判断是否越界。

string s; reverse(s.begin(), s.end()); 将字符串s进行翻转。

进制转换可以全部转换成十进制，然后再用求余和求商将其转换成目标进制。

数组下标作为一个标识是非常好用的，可以当作地址，节点的序号等等等等。

模拟栈，一个序列入栈，给出一个出栈序列，问可不可能。就从头入栈，然后比较出栈元素和栈首，当相等就出栈，然后继续比较；不等就继续入栈，同时判断栈是否溢出。

输出时注意精度和速度

当对输出浮点数的精度有要求时，用printf("%.2f",a)最为方便。

cout不加控制符的话无法控制精度。

在其他情况下，则是使用cout更加方便。

另外，PAT中有时会出现运行超时，这是因为cin和cout不够快速，这时候就应该使用

int id,score;

char c[80];

scanf("%d %s %d",&id,name,&score);

printf("%06d %s %d\n",id,name,score);

注意：

使用scanf比较蛋疼的地方在于，存储字符串就只能使用char数组了，而且比较还要使用strcmp；

注意输入%s到char数组时，不需要取地址，因为数组名本身就是地址。

使用printf时，应该灵活地使用格式控制。

读取输入测试例的方式

如果测试例用空格或回车分隔，那么可以直接使用cin。   
如果要求逐个读取字符，并以回车’\n’结束，那么可以使用getchar（或cin.get）函数

不能直接采用

while( cin>>c )

因为cin会忽略回车和空格，这将导致不能读取输入。

另外，如果是将cin和getline搭配使用，要格外注意回车'\n'。

例如，有若干行数据等待读入，第一行用cin>>读入，但由于cin会忽略回车，所以如果在第二行使用getline（cin,s）的话会发生错误。原因是getline（cin,s）读取的只是第一行末尾的'\n'。   
正确的做法是

cin>>a; getchar(); getline(cin,s);

即用一个getchar()将那个'\n'吸收掉。

不知为何，PAT里判断c是否是小写字母，只能用STL里的islower(c)函数，而不能用if(c>=’a’ && c<=’z’) 的方法。   
所以尽量使用STL中的标准库函数。

素数表用筛素数法建立，建素数表时应该把范围确定在尽量小的范围尽量避免时间超界

从2开始筛素数，范围内筛掉后朝前找到的第一个未被标记是素数的数就是当前最小的素数，然后继续筛就好。

如果答案集合在一个大的集合中的一个子集中，比如字符集中的英文字母或者阿拉伯数字，应该加上一次判断是否在需要的范围内。

声明一个指针让它为空。

node\* root = NULL;

哈希表冲突解决方法

Quadratic probing（二次方探查法） (with positive increments only) is used to solve the collisions

(key + step \* step) % size step步长从1加到size-1

大小写字符转换

大写转小写 + 32

小写转大写 -32

在题目内容是序列时，应该考虑排序是否能给解题带来便利。

将向量进行长度初始化

vector<int> father,isRoot;

father.resize(n + 1);

isRoot.resize(n + 1);

运算符重载

bool operator < (const node &a) const {

        return (cnt != a.cnt) ? cnt > a.cnt : value < a.value;

    }

struct node {

    int value, cnt;

    node(int a, int b):value(a), cnt(b){}

    bool operator < (const node &a) const {

        return (cnt != a.cnt) ? cnt > a.cnt : value < a.value;

    }

};

拓扑排序：找没有前驱节点的将其删除，更新图，再次找，再次删除。直到图中没有节点。